Pulsassosiasjon og geolokalisering med bruk av radarpulsers ankomsttid til to plattformer i parallell bevegelse

Masteroppgave

Magnus Baksaas

Fysisk Institutt

UNIVERSITETET I OSLO

25. mai 2015

Forord

Denne rapporten er skrevet i forbindelse med min masteroppgave, og er avslutningen på mitt masterstudium i Elektronikk og datateknologi, studieretning Kybernetikk, ved Fysisk institutt på Universitetet i Oslo.

Oppgaven ble formulert av Forsvarets forskningsinstitutt (FFI), avdeling for Cybersystemer og elektronisk krigføring, i samråd med Universitetssenteret på Kjeller (UNIK). Oppgaven tar for seg å assosiere pulser fra flere samtidige radarer slik at disse kan geolokaliseres ved kun å bruke tidsdifferansen mellom pulsenes ankomsttid (TDOA) til to plattformer i bevegelse.

Jeg vil rette en spesiell takk til veilederne mine, forsker Tore Smestad ved FFI og professor Oddvar Hallingstad ved UNIK, for god hjelp under arbeidet med oppgaven.

Kjeller, mai 2015

Magnus Baksaas

Sammendrag

Denne oppgaven har tatt for seg å assosiere pulser fra flere samtidige radarer slik at disse kan geolokaliseres ved kun å bruke tidsdifferansen mellom pulsenes ankomsttid (TDOA) til to plattformer i bevegelse. Tidligere analyser tyder på at TDOA er tilstrekkelig for å finne riktige posisjoner, men det har ikke vært analysert hvilke forhold som avgjør hvor mange radarer som lar seg geolokalisere. Det finnes mange anvendelsesområder for å geolokalisere radarer ved bruk av TDOA, men i denne oppgaven er det kun sett på geolokalisering av skip med navigasjonsradarer fra to små koordinerte ubemannede fly (UAS) eller fra to «tandem-satellitter».

I oppgaven er det utviklet en algoritme for å analysere problemstillingen. Algoritmen assosierer pulser fra flere samtidige radarer og beregner TDOA-er. Deretter assosieres TDOA-er fra samme radarer slik at disse kan geolokaliseres. Algoritmen er basert på en «greedy»-filosofi, for å redusere beregningstid og antall estimater. Til å generere ankomsttider er det laget en simulator, hvor det simuleres to plattformer i bevegelse og et antall roterende radarer som sender ut pulser basert på ekte målinger. For begge anvendelsesområder antas det at jorden er flat, ettersom dette fører til enklere formler, mindre regnetid og mer tid til de egentlige problemstillingene.

For å undersøke hvilke forhold som avgjør hvor mange radarer som lar seg geolokalisere ved bruk av den utviklede algoritmen, er ulike simulerte datasett analysert ved Monte Carlo simuleringer. For å få et bilde av estimeringsnøyaktigheten, og hvilke forhold som påvirker denne, er resultatene fra Monte Carlo simuleringen sammenliknet med hva som er oppnåelig ifølge Cramér Rao Lower Bound (CRLB).

Oppgaven viser at det er tilstrekkelig å bare bruke TDOA for å assosiere pulser fra flere samtidige radarer, slik at disse kan geolokaliseres. Det avgjørende for at radarene skal la seg geolokalisere, er at radarene ligger slik til at sensorene mottar TDOA-er fra radarene som ikke assosieres til andre radarer. Analysene viser at dersom dette er tilfelle, gir algoritmen estimater av radarposisjonene med en estimeringsnøyaktighet som er i overensstemmelse med CRLB. Derimot hvis TDOA-ene overlapper og assosieres til feil radarer, gir algoritmen ofte et felles estimat for begge radarene eller at bare en av radarene estimeres. I disse tilfellene kan det trolig oppnås bedre resultater ved å benytte andre tilnærminger enn greedy-filosofien. I oppgaven er det kun identifisert et tilfelle hvor flere radarer ikke lar seg geolokalisere ved å bare bruke TDOA, nemlig når disse lå så tett at deres feilellipser overlapper og det vil da ikke være mulig å skille radarene ved å kun bruke TDOA.

Innholdsfortegnelse

Figurliste

Tabelliste

Nomenklatur

CEP	Circular Error Probable
CRLB	Cramér-Rao Lower Bound
ESM	Electromagnetic Support Measures
FFI	Forsvarets forskningsinstitutt
KTM	Koordinattransformasjonsmatrise
LOS	Line Of Sight
PRF	Pulsrepetisjonsfrekvens
PRI	Pulsrepetisjonsintervall
RKM	Retningskosinmatrise
RMS	Root Mean Square
TDOA	Differanse i ankomsttid (Time Difference Of Arrival)
TOA	Ankomsttid (Time Of Arrival)
UAS	Ubemannet fly (Unmanned Aircraft Systems)

1 Innledning

1.1 Bakgrunn

Forsvarets forskningsinstitutt (FFI) har gjennom mange år undersøkt ulike metoder for å geolokalisere radarer ved hjelp av sensorer som mottar utsendte radarpulser. Tidsdifferanse (TDOA) er en metode som kan gi rask og nøyaktig geolokalisering av radarer. Den krever imidlertid tre sensorer for å gi en momentan lokalisering. Med to plattformer i bevegelse kan en også finne radarers posisjon, men da etter en viss tid. Tidligere undersøkelser tyder på at radarpulsenes ankomsttid i to plattformer i «tandem-bevegelse» er tilstrekkelig for å assosiere mottatte pulser fra mange radarer og samtidig geolokalisere disse. Mulige anvendelser kan være lokalisering av skip med navigasjonsradarer fra to små koordinerte ubemannede fly (UAS) eller fra to «tandem-satellitter». Det er interessant å få en god forståelse av egenskapene til denne metoden i disse to anvendelsene.

FFI begynte for alvor å undersøke mulighetene for geolokalisering av radarer fra to plattformer i parallell bevegelse med masteroppgaven [1]. I oppgaven ble det undersøkt gjennomførbarheten og nøyaktigheten ved geolokalisering av navigasjonsradarer fra to ESM-sensorer ombord på to bevegende plattformer vha. geolokaliseringsmetodene TDOA, skannfase og en kombinasjon av disse. Det ble i oppgaven kun utført teoretiske analyser ved Cramér-Rao Lower Bound (CRLB), men geolokaliseringsnøyaktigheten ble vurdert som så god at FFI ønsket flere analyser. Masteroppgaven resulterte i rapporten [2] som omhandler geolokalisering av flere navigasjonsraderer fra to fly i bevegelse. Rapporten omhandler ulike metoder for å assosiere TDOA fra flere radarer og algoritmer for å geolokalisere disse. FFI ser nå på mulige anvendelser for geolokalisering av radarer ved bruk av TDOA. Til sommeren skal FFI sende opp to UAS med hver sin sensor for mottak av radarpulser. Her tenkes det å geolokalisere skip vha. TDOA og skannfase. FFI har også innledet et samarbeid med Nederland om å se på muligheten for å benytte TDOA mellom to satellitter for å geolokalisere navigasjonsradarer. I forbindelse med dette samarbeidet ble notatet [3] skrevet. Notatet omhandler noen geometriske betraktninger og en mulig oppnåelig estimeringsnøyaktighet. Det bemerkes at det å finne den samme pulsen i begge satellittene for å beregne TDOA og det å assosiere TDOA fra samme radar i løpet av en hel overflyvning kan være mulige problemer. I notatet ble estimeringsnøyaktigheten vurdert som så god at FFI ønsker å gå videre med problemstillingen. Å assosiere TDOA fra samme radar vil være relevant i mange sammenhenger, deriblant de to nevnte anvendelsesområdene, og FFI ønsker flere analyser på problemstillingen.

Det finnes lite tilgjengelig informasjon som omhandler assossiasjon og geolokalisering av radarer vha. TDOA mellom to plattformer i bevegelse. I de artikler og rapporter som finnes, ser det ut til at forfatterne er svært tilbakeholdene med informasjon. [4] omhandler passiv geolokalisering og følging av et ukjent antall radarer. Artikkelen tar utgangspunkt i flere UAS-er med sensorer for mottak av radarpulser, men det er ikke nevnt hvor mange. I [5] geolokaliseres radarer fra to UAS-er, og de analyserer to ulike metoder for følging. Men artikkelen ser bare på tilfeller med en radar. [6] omhandler en rekursiv estimator for geolokalisering av radarer vha. TDOA målinger fra to UAS-er. Også i denne artikkelen ses det kun på en radar.

1.2 Oppgaven

Oppgaven går ut på å etablere forståelse av problemstillinger knyttet til mulige realiseringer av metoden for pulsassosiasjon og geolokalisering av radarer fra to plattformer i parallell bevegelse ved å kun bruke TDOA. Det finnes mange andre metoder for å skille pulsene fra ulike radarene, som ved f.eks. radarens frekvens, pulsrepetisjonsintervall (PRI), rundetider og andre karakteristiske parametere. I oppgaven skal det ses på hvor godt pulsassosiasjonen og geolokaliseringen kan utføres når disse parameterne ikke er tilgjengelig og kun pulsers ankomsttid kan benyttes. Dette innebærer å videreutvikle eksisterende programmer og metoder for å gjøre simuleringer, samt analysere måledata der slike måtte være tilgjengelige. Analysene bør innebefatte geometrier og parameterverdier som er realistiske for begge de nevnte mulige anvendelser. Statistikk hvor «slengere» og andre ikke-gaussiske fordelinger inngår bør også være en del av problemstillingene. I oppgaveteksten ble følgende arbeidsmomenter nevnt:

A. Beskrive problemstillingen med støtte i egnede fagområder og tidligere arbeider, og lage simuleringsprogram for å støtte analyser av problemstillingene.
B. Kartlegge sammenhengen mellom de viktigste parameterne som bestemmer geolokaliseringsnøyaktigheten, så som geometri, målenøyaktighet osv. Her tenkes CRLB å være en egnet metode. Resultatene bør understøttes med simulering av geolokalisering og reelle data hvis mulig.
C. Utvikle og teste en eller flere metoder for å assosiere pulser fra samtidige radarer for å geolokalisere disse. Bruke den mest egnede metoden for å finne sammenhenger mellom de viktigste forhold som avgjør hvor mange samtidige radarer som lar seg geolokalisere. Bør demonstreres med reelle data hvis mulig.
D. Bruke analyser og verktøy fra punktene B og C til å gi et anslag av hva en kan oppnå med aktuelle geometrier og parameterverdier i de to nevnte anvendelsene med satellitt og UAS.
E. Oppsummere avklarte egenskaper og problemer til denne geolokaliseringsmetoden samt gi eventuelle råd om løsninger, implementasjon og videre arbeid.

1.3 Rapportens oppbygning

Oppgaven er delt inn i 11 kapitler. Kapittel 2 inneholder en beskrivelse problemstillingen og de to scenarioene som benyttes i oppgaven. Kapittel 3 inneholder matematiske modeller som benyttes i oppgaven og kapittel 4 inneholder simuleringsprogram for å generere ankomsttider. Kapittel 5 beskriver prinsippet for hvordan TDOA beregnes mellom pulser og hvordan radarer kan geolokaliseres ved bruk av TDOA, og kapittel 6 beskriver en egenutviklet algoritme for å assosiere pulser fra samtidige radarer og geolokalisering av disse. I kapittel 7 testes og undersøkes det hvor godt algoritmen assosierer pulser fra radarer og hvor godt radarene geolokaliseres. Kapittelet begynner med å undersøke oppnåelig estimeringsnøyaktighet ved CRLB, for så å teste algoritmen på ulike testoppsett. I kapittel 8 testes og undersøkes det hvor tett to radarer kan ligge og at algoritmen allikevel skal klare å geolokalisere radarene. I kapittel 9 testes og undersøkes det hvor godt algoritmen assosierer pulser fra radarer og hvor godt radarene geolokaliseres når målingene inneholder slengere. Kapittel 10 diskuterer analysene og resultatene fra de tre foregående kapitlene, ulike algoritmeaspekter diskuteres og forslag til videre arbeid. Kapittel 11 inneholder en konklusjon av arbeidet.

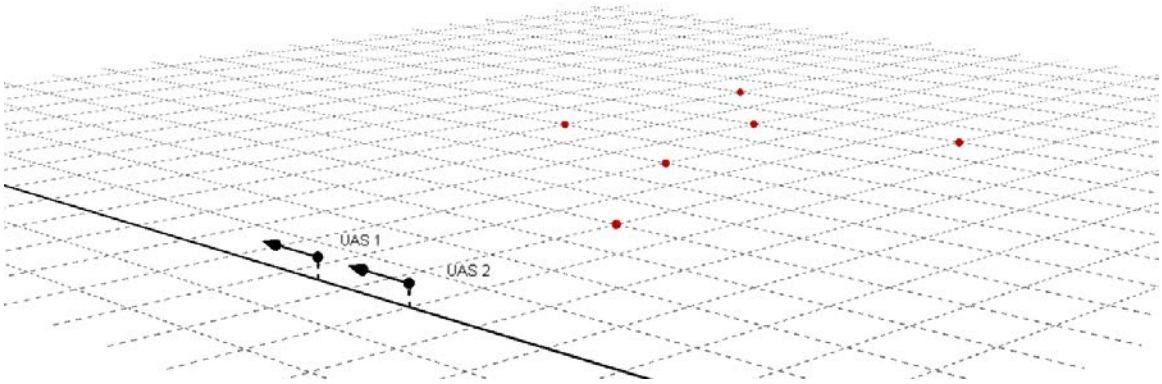
2 Scenarioer

Geolokalisering av radarer ved bruk av pulser ankomsttid til to plattformer kan benyttes i mange sammenhenger, men oppgaven begrenser seg til to anvendelsesområder; geolokalisering av skip med navigasjonsradarer fra to små koordinerte UAS-er og fra to små tandem-satellitter. Under følger en nærmere beskrivelse av de to scenarioene som benyttes i rapporten.

2.1 Scenario med to UAS-er

I første scenario lokaliseres skip med navigasjonsradarer fra to små koordinerte UAS-er, som kan være svært anvendelig for kystovervåkning. UAS-ene vil være utstyrt med hver sin sensor for mottak av radarpulser. I scenarioet vil to UAS-er fly langs en rett trajektorie med en konstant hastighet og i en konstant høyde. I [1] ble det konkludert med at nøyaktigheten ble best når UAS-ene flyr etter hverandre og når radarene som skal geolokaliseres ligger ut mot en av sidene.

UAS-ene vil være utstyrt med hver sin dipolantenne for mottak av radarpulser. Dipolantennen vil være montert slik at dipolene ligger parallelt med flyretningen til flyene. Pga. dipolantennens virkemåte vil antennen da kunne motta pulser fra radarer i nesten alle retninger, bortsett fra fremover og bakover retningen til UAS-ene. UAS-ene vil typisk fly i en høyde på rundt 1500 fot (500 m benyttes i oppgaven) og ha en hastighet på 70-100 km/t (20 m/s benyttes i oppgaven). Avstanden ut til radarene vil typisk ligge i området 10-50 km. I scenarioet antas det at radarene som skal geolokaliseres er navigasjonsradarer (for mer informasjon om navigasjonsradarer, se appendiks A). Radarene vil ha ulik rotasjonshastighet og ha ulik pulsrepetisjonsfrekvens (PRF). Dette resulterer i at sensorene noen ganger mottar pulser fra en og en radar av gangen, mens andre ganger mottar pulser fra flere radarer samtidig. Ettersom UAS-ene flyr i lav høyde over jordoverflaten er jorden praktisk talt flat i dette scenarioet, og derfor benyttes det flat jord i oppgaven. Figur 2.1 viser et typisk scenario som analyseres i oppgaven, hvor to UAS-er følger en rett bane og hvor det ligger en rekke skip med navigasjonsradarer ut mot en av sidene av flybanen.



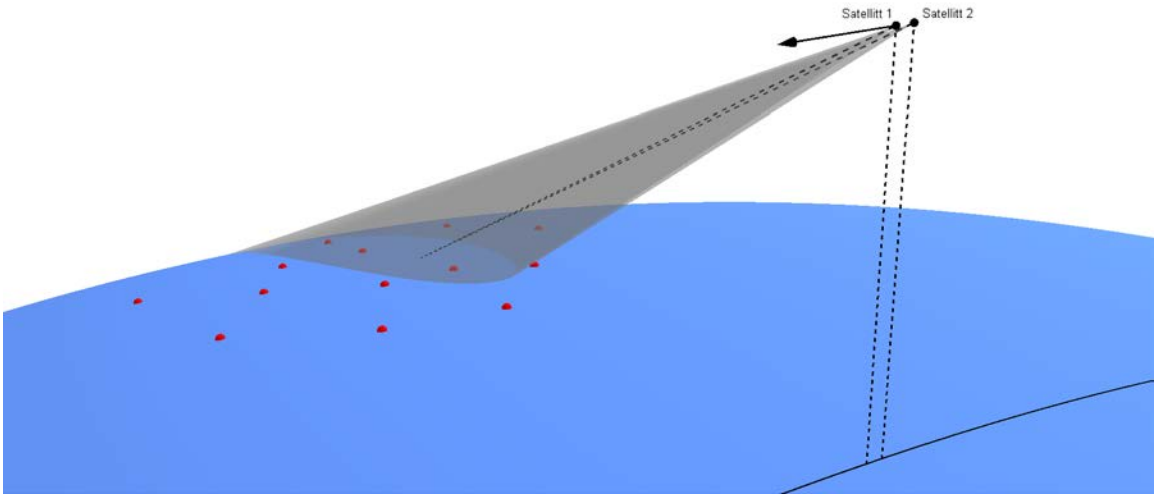
Figur 2.1: Scenario med to UAS-er som flyr langs en rett trajektorie og seks skip med navigasjonsradarer (røde kuler).

UAS-ene flyr med forholdsvis lav hastighet, og vil derfor motta mange pulser fra radarene i løpet av en hel overflyvning. For å geolokalisere radarer ved TDOA, bør vinkelendringen til sensorene, sett fra radarene, være størst mulig for at geolokaliseringsnøyaktigheten skal bli best mulig. For å redusere datamengden er det derfor valgt å simulere data fra flere korte måleintervall, samtidig som måleintervallene er fordelt utover tiden slik at geolokaliseringsnøyaktigheten blir best mulig.

2.2 Scenario med to satellitter

I andre scenario ligger to satellitter i den samme polare satellittbanen og geolokaliserer navigasjonsradarer til skip. Scenarioet er svært relevant sett med norske øyne, ettersom satellittene vil fly over Barentshavet og Norskehavet ved de fleste omløpene. Skipenes posisjon estimert vha. TDOA kan da komplementere annen informasjon som f.eks. AIS (Automatic Information System – selvrapportering av posisjon, kurs, hastighet, osv.) og gi et bilde av hvilke skip som oppholder som i Norske farvann.

Satellittene vil ligge i en lav polarbane, omtrent 600 km over jordoverflaten. Avstanden mellom satellittene vil være 30 km [3], og banehastigheten vil være 27 000 km/t (7500 m/s). Begge satellittene vil være utstyrt med hver sin retningsbestemte antenne for mottak av radarpulser. Lobebredden til antennene vil i oppgaven være 10° i både asimut og elevasjon. For TDOA er det en forutsetning at begge sensorene mottar de samme pulsene. Satellittene bør derfor kunne roteres slik at hovedlobene dekker det samme området nede på jordoverflaten. For at satellittene skal kunne geolokalisere skip i et størst mulig område, vil det være naturlig at hovedlobene peker ut mot horisonten. Satellittene vil da dekke området fra rundt 1000 km og ut til jordkrumningen på 2700 km, og området vil være rundt 350 km bredt. Ettersom satellittene ligger så høyt over jordoverflaten vil de elektromagnetiske pulsene fra radarene propagere i den rette linjen mellom radarene og satellittene. Satellittene vil derfor ikke kunne motta pulser fra radarer som ligger bak jordkrumningen. Figur 2.2 viser to satellitter som ligger i samme bane med en fast innbyrdes avstand og en rekke navigasjonsradarer nede på jordoverflaten. Begge satellittene er orientert slik at deres hovedlober dekker det samme området nede på jordoverflaten. Under satellittene er det tegnet opp trajektorien satellittene følger.



Figur 2.2: Scenario med to satellitter med hovedlober (grå kjegler) som dekker et område med flere navigasjonsradarer (røde kuler).

I oppgaven ligger hovedutfordringen i å assosiere pulser fra samtidige radarer for å kunne geolokalisere disse. Derfor antas det at jorden er flat i satellitt-scenarioet. De samme metodene for å assosiere pulser og geolokalisere radarer kan da benyttes i begge scenarioer.

3 Matematiske modeller

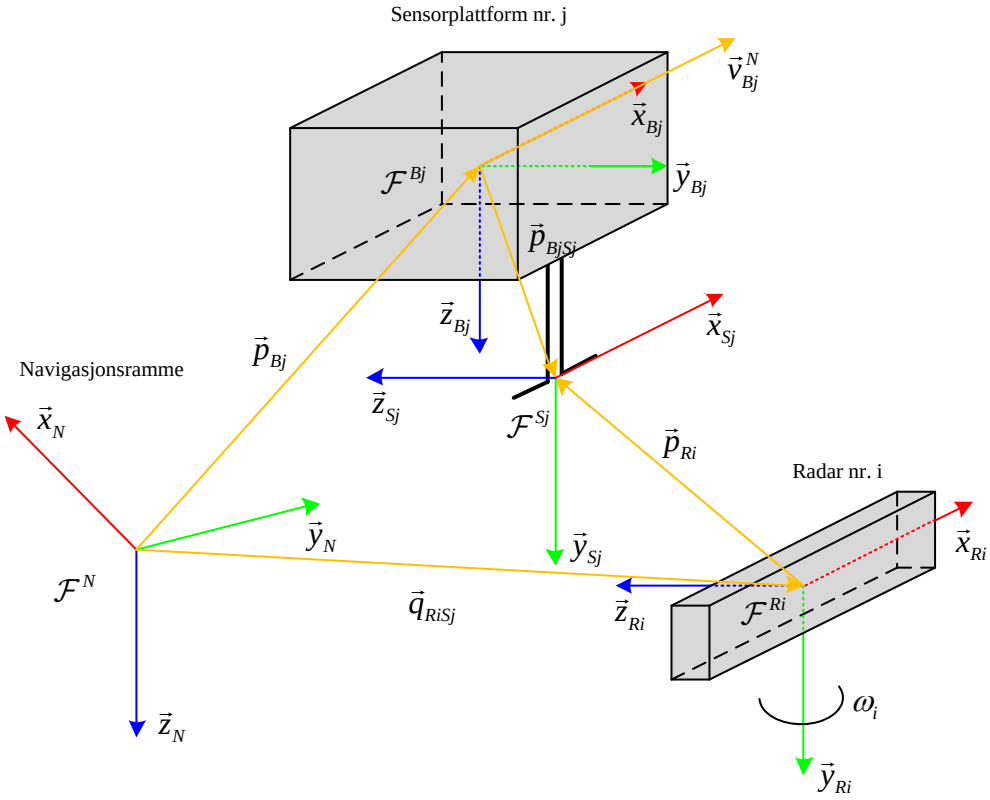
Kapittelet inneholder matematiske modeller av to sensorplattformer i bevegelse og et antall roterende navigasjonsradarer. I simuleringen av UAS-scenarioet, benyttes det kunnskap om sensorenes og radarenes strålingsdiagram for å bestemme om sensorene kan motta pulsene. Tidligere analyser ved FFI viser at satellitter vil kunne motta pulser sendt ut fra navigasjonsradarer. Trolig vil satellittene bare motta pulser som sendes fra radarenes hovedlober og som mottas av satellittenes hovedlober. For å spare regnetid i simuleringene, mottar derfor satellittene kun pulser som sendes ut fra radarenes hovedlober og som mottas av satellittenes hovedlober. Som beskrevet i kapittel 2 er det benyttet flat jord for begge anvendelsesområdene. Videre antas det at pulsene fra radarene propagerer i rette linjer. Forenklingene fører til enklere formler, mindre regnetid og mer tid til de egentlige problemstillingene.

Kapittelet er delt inn i to avsnitt; avsnitt 3.1 inneholder matematiske modeller for UAS-scenarioet, og avsnitt 3.2 inneholder matematiske modeller for satellitt- scenarioet. Modellene som beskrives i kapittelet benyttes primært til å simulere radarpulsers ankomsttid i to plattformer beskrevet i kapittel 4, men vil også ligge til grunn når sensorenes posisjon og orientering omtales senere i rapporten. Notasjon og et matematisk grunnlag er beskrevet i appendiks B.

3.1 Matematiske modeller for UAS-scenario

3.1.1 Definisjon av rammer og vektorer

For å beskrive posisjon og orientering til sensorplattformene og navigasjonsradarene, trengs det et sett med rammer og vektorer mellom dem. Figur 3.1 viser en oversikt over rammer for å beskrive en sensorplattform og en roterende radar, samt vektorene mellom rammene. Definisjonene av rammene og vektorene følger etter figuren.



Figur 3.1: Rammer og vektorer for UAS-scenario.

3.1.1.1 Rammer

Til å bestemme posisjonene til UAS-ene og til navigasjonsradarene, trengs det en navigasjonsramme. Innfører navigasjonsrammen $\mathcal{F}^N = \{O_N; \vec{x}_N, \vec{y}_N, \vec{z}_N\}$, hvor x-aksen alltid peker mot nord, y-aksen alltid mot øst og z-aksen alltid ned. Enheten langs de tre aksene er meter.

På jorden befinner det seg til enhver tid to sensorplattformer, begge med hver med sin bodyramme $\mathcal{F}^{Bj} = \{O_{Bj}; \vec{x}_{Bj}, \vec{y}_{Bj}, \vec{z}_{Bj}\}$ for sensorplattform $j = 1,2$. Bodyrammene har origo i massesenteret til sensorplattformen; og er orientert slik at x-aksen alltid peker rett frem i sensorplattformen, y-aksen alltid ut mot høyre og z-aksen alltid rett ned. Enheten langs de tre aksene er meter. Sensorplattformene er utstyrt med hver sin dipolantenne. Dipolantenna beskrives i sensorrammen $\mathcal{F}^{Sj} = \{O_{Sj}; \vec{x}_{Sj}, \vec{y}_{Sj}, \vec{z}_{Sj}\}$. Dipolantenna og dermed sensorrammen står fast i forhold til bodyrammen. Sensorrammen har origo midt mellom dipolene; og er orientert slik at x-aksen alltid peker parallelt med x-aksen til bodyrammen og y-aksen parallelt med z-aksen til bodyrammen. z-aksen står normalt på x- og y-aksen, slik at rammen danner et høyrehåndssystem.

På jordoverflaten befinner det seg et uvisst antall skip med navigasjonsradarer. I denne oppgaven antas det at skipene står i ro, uten å rotere. Det er derfor ikke nødvendig å beskrive skipene. Hver navigasjonsradar beskrives derfor av hver sin radarramme $\mathcal{F}^{Ri} = \{O_{Ri}; \vec{x}_{Ri}, \vec{y}_{Ri}, \vec{z}_{Ri}\}$ for $i = 1,2,\dots,I$, hvor I er antall navigasjonsradarer. Radarrammen har fast posisjon i navigasjonsrammen, men vil være orientert slik at y-aksen alltid peker parallelt med z-aksen i navigasjonsrammen.

Navigasjonsradarene vil rotere om y-aksen sin. x-aksen peker parallelt med linjen gjennom antenneelementene og z-aksen i retning av hovedloba til navigasjonsradaren.

3.1.1.2 Posisjonsvektorer

Posisjonen til de ulike rammene bestemmes ved hjelp av posisjonsvektorene. Posisjonen til bodyrammen på sensorplattform j vil være gitt av posisjonsvektoren $\vec{p}_{Bj}$, og vil typisk bli målt i navigasjonsrammen. Posisjonen til sensorramme j vil være gitt av posisjonsvektoren $\vec{p}_{BjSj}$, og vil være kjent i bodyramme j. Posisjonen til navigasjonsradar i vil være gitt av posisjonsvektoren $\vec{p}_{Ri}$, og vil kun være kjent i en simuleringssituasjon.

3.1.1.3 Propagasjonsvektorer

Pulser som sendes ut av navigasjonsradar i og mottas av sensor j vil propagere langs vektoren $\vec{q}_{RiSj}$, og vil kun være kjent i en simuleringssituasjon. Vinklene mellom propagasjonsvektoren og radarrammen vil bestemme hvor stort gain radaren har, mens vinklene mellom propagasjonsvektoren og sensorrammen bestemmer hvor stort gain dipolantennen på sensorplattformen har. Lengden til propagasjonsvektoren bestemmer, sammen med utsendelsestidspunktet til en puls, ankomsttiden til pulsen i sensorplattformen.

3.1.1.4 Hastighetsvektorer

Sensorplattformene vil bevege seg i navigasjonsrammen. Hastigheten til sensorplattform j beskrives ved hastighetsvektoren $\vec{v}_{Bj}^{N}$, hvor posisjonsendringen er sett fra navigasjonsrammen.

3.1.2 Beregning av vektorer og koordinattransformasjonsmatriser

Pulssimulatoren har en liste over utsendelsestidspunkter til pulser. Det enkleste er da å oppdatere UAS-enes posisjon og orientering, og radarens orientering i disse tidspunktene. Det forutsettes da at UAS-enes hastighet og radarenes rotasjonshastigheter er konstant mellom utsendelsestidspunktene.

3.1.2.1 Koordinattransformasjonsmatriser

Koordinattransformasjonsmatrise fra bodyramme på sensorplattform j til navigasjonsrammen

Antar i simuleringen at sensorplattformene beveger seg rett nordover, uten å rotere eller endre høyde. Da vil navigasjonsrammen og sensorrammen ha lik orientering, og koordinattransformasjonsmatrisen (KTM) fra bodyramme j til navigasjonsrammen blir

$$R_{Bj}^{N} = I. \tag{3.1}$$

Koordinattransformasjonsmatrise fra sensorramme til bodyramme på sensorplattform *j*

Sensorrammen er orientert slik at x-aksen står parallelt med x-aksen til bodyrammen på sensorplattform j. Videre er sensorrammen orientert slik at y-aksen peker parallelt med z-aksen til bodyrammen. z-aksen står slik at sensorrammen danner et høyrehåndssystem. KTM fra sensorrammen til bodyrammen blir da

$$R_{Sj}^{Bj} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{3.2}$$

Koordinattransformasjonsmatrise fra radarramme *i* til navigasjonsrammen

Radarrammen er orientert slik at y-aksen alltid er parallell med z-aksen i navigasjonsrammen. Videre roterer radarrammen om y-aksen sin med en konstant rotasjonshastighet ω_i. Radarrammen kan beskrives ved at navigasjonsrammen først roteres 90° om x-aksen, og deretter en vinkel ψ_i om y-aksen. KTM fra radarramme i til navigasjonsrammen beskrives ved

$$R_{Ri}^{N} = [\boldsymbol{R}_{NRi}]^{N} = R_x\left(\frac{\pi}{2}\right) R_y(\psi_i), \tag{3.3}$$

hvor ψ_i er vinkelen det er rotert om y-aksen. Antar at radaren roterer med en konstant vinkelhastighet dvs. at $\omega_i(t)$ er konstant. Vinkelen til radaren ved tidspunkt t_k kan da beregnes rekursivt ved

$$\psi_i(t_k) = \psi_i(t_{k-1}) + \omega_i \cdot (t_k - t_{k-1}), \tag{3.4}$$

hvor $\psi_i(t_{k-1})$ er vinkelen mellom x-aksen i radarrammen og x-aksen i navigasjonsrammen i tidspunkt t_{k-1}. Ved å sette (3.4) inn i formel (3.3), kan KTM beregnes ved

$$\begin{aligned} R_{Ri}^{N}(t_k) &= R_x\left(\frac{\pi}{2}\right) R_y\big(\psi_i(t_{k-1}) + \omega_i \cdot (t_k - t_{k-1})\big) \\ &= R_x\left(\frac{\pi}{2}\right) R_y\big(\psi_i(t_{k-1})\big) R_y\big(\omega_i \cdot (t_k - t_{k-1})\big). \end{aligned} \tag{3.5}$$

I første iterasjon ($k = 0$) antas det at KTM er

$$R_{Ri}^{N}(t_0) = R_x\left(\frac{\pi}{2}\right) R_y\big(\psi_i(t_0)\big). \tag{3.6}$$

I neste iterasjon ($k = 1$) blir KTM

$$\begin{aligned} R_{Ri}^{N}(t_1) &= R_x\left(\frac{\pi}{2}\right) R_y\big(\psi_i(t_0)\big) R_y\big(\omega_i \cdot (t_1 - t_0)\big) \\ &= R_{Ri}^{N}(t_0) R_y\big(\omega_i \cdot (t_1 - t_0)\big). \end{aligned} \tag{3.7}$$

For generelle k kan KTM beregnes rekursivt ved

$$R_{Ri}^{N}(t_k) = R_{Ri}^{N}(t_{k-1}) R_y\big(\omega_i \cdot (t_k - t_{k-1})\big). \tag{3.8}$$

3.1.2.2 Posisjonsvektorer

Posisjonsvektor til sensorplattform *j*
Hastigheten til sensorplattform j vil være en posisjonsendring i navigasjonsrammen, men vil typisk måles i sensorplattformen; og betegnes derfor $\underline{v}_{Bj}^{NBj}(t)$. Hastigheten til sensorplattform j antas å være konstant, dvs. at $\underline{v}_{Bj}^{NBj}(t)$ er konstant. Hastigheten til sensorplattform j, sett fra navigasjonsrammen bestemmes ved

$$\underline{v}_{Bj}^{N} = R_{Bj}^{N}\underline{v}_{Bj}^{NBj}, \tag{3.9}$$

hvor $\underline{v}_{Bj}^{NBj}$ er hastigheten til sensorplattformen, derivert fra navigasjonsrammen og sett fra sensorplattformen. Posisjonen til sensorrammen beregnes rekursivt ved

$$\begin{aligned}\underline{p}_{Bj}^{N}(t_k) &= \underline{p}_{Bj}^{N}(t_{k-1}) + \underline{v}_{Bj}^{N}\cdot(t_k - t_{k-1}) \\ &= \underline{p}_{Bj}^{N}(t_{k-1}) + R_{Bj}^{N}(t_k)\underline{v}_{Bj}^{NBj}\cdot(t_k - t_{k-1}),\end{aligned} \tag{3.10}$$

hvor $\underline{p}_{Bj}^{N}(t_{k-1})$ posisjonen ved forrige tidspunkt.

Posisjonsvektor til sensor på sensorplattform *j*
Antar at sensoren står fast i sensorplattform j. Posisjonsvektoren til sensorrammen beregnes ved

$$\underline{p}_{Sj}^{N}(t_k) = \underline{p}_{Bj}^{N}(t_k) + R_{Bj}^{N}\underline{p}_{BjSj}^{Bj}. \tag{3.11}$$

Posisjonsvektor til radarramme *i*
Ettersom navigasjonsradarene har fast posisjon vil posisjonsvektoren til radarrammen, $\underline{p}_{Ri}^{N}$, være konstant.

3.1.2.3 Propagasjonsvektorer

Propagasjonsvektor i navigasjonsrammen
Propagasjonsvektoren sett fra navigasjonsrammen benyttes til å beregne tiden det tar for en puls å propagere fra radaren til sensoren, og beregnes ved

$$\underline{q}_{RiSj}^{N}(t_k) = \underline{p}_{Sj}^{N}(t_k) - \underline{p}_{Ri}^{N} = \underline{p}_{Bj}^{N}(t_k) + R_{Bj}^{N}\underline{p}_{BjSj}^{Bj} - \underline{p}_{Ri}^{N}. \tag{3.12}$$

Propagasjonsvektor i radarramme *i*
For å beregne vinklene mellom propagasjonsvektoren og radarramme i, må propagasjonsvektoren representeres i radarramme i. Propagasjonsvektoren sett fra radarramme i beregnes ved

$$\underline{q}_{RiSj}^{Ri}(t_k) = R_{N}^{Ri}(t_k)\underline{q}_{RiSj}^{N}(t_k). \tag{3.13}$$

Propagasjonsvektor i sensorramme *j*
For å beregne vinklene mellom propagasjonsvektoren og sensorramme j, må propagasjonsvektoren representeres i sensorramme j. Propagasjonsvektoren sett fra sensorramme j beregnes ved

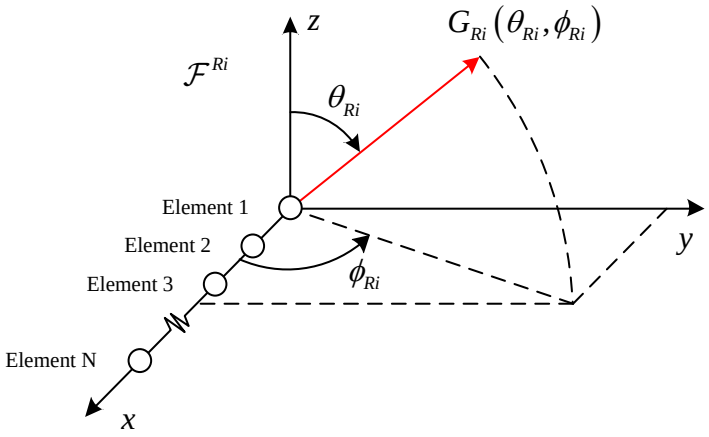
$$\underline{q}_{RiSj}^{Sj}(t_k) = R_{N}^{Sj}\underline{q}_{RiSj}^{N}(t_k). \tag{3.14}$$

3.1.3 Propagasjon av radarpulser

3.1.3.1 Strålingsdiagram og forsterkning i retning av propagasjonsvektor

Strålingsdiagram for radar i

Navigasjonsradarer består av et array av strålingskilder, ofte uniformt fordelt utover en rett linje. Strålingskildene vil typisk ligge på et jordplan for å begrense sidelober og bakloben. Radaren i oppgaven blir modellert ved å plassere N strålingskilder langs x-aksen i radarrammen $\mathcal{F}^{Ri}$, se Figur 3.2.

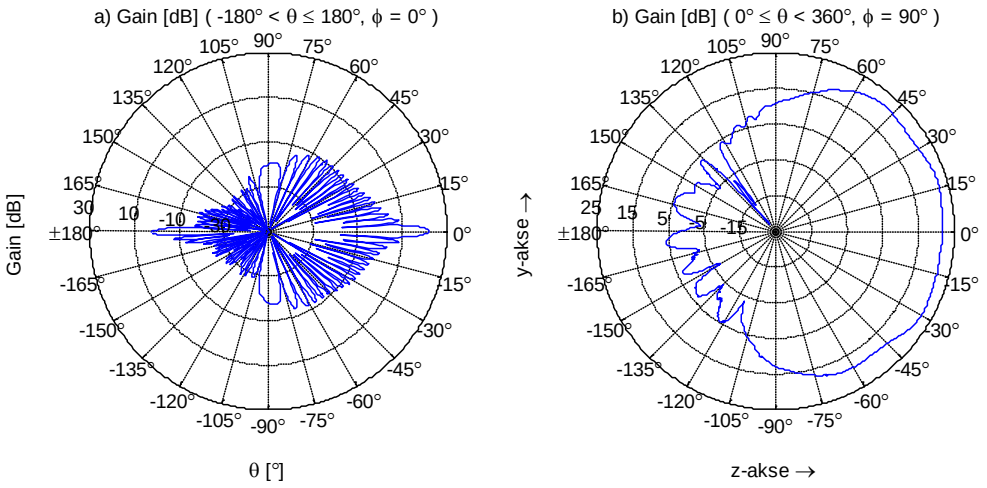


Figur 3.2: Navigasjonsradar med N strålingskilder uniformt fordelt langs x-aksen i radarrammen $\mathcal{F}^{Ri}$.

Radarens forsterkning beregnes ved formel (1.17) i [7]

$$G_{Ri}(\theta_{Ri},\phi_{Ri}) = \left| G(\theta_{Ri},\phi_{Ri}) \frac{\sum_{n=1}^{N} A_n \exp(jk_0 na \sin\theta_{Ri} \cos\phi_{Ri})}{\sqrt{\sum_{n=1}^{N} |A_n|^2}} \right|, \qquad (3.15)$$

hvor $G(\theta_{Ri},\phi_{Ri})$ er gain til hver enkelt strålingskilde, A_n er vekting av strålingskildene, k_0 er vinkelbølgetallet, og a er avstanden mellom strålingskildene. Figur 3.3 viser strålingsdiagrammet til navigasjonsradarene som simuleres i simulatoren. Radaren består av 35 strålingskilder, hvor avstanden mellom dem er en halv bølgelengde og hver strålingskilde vektes likt. Dette gir en lobebredde på 3°, som tilsvarer en typisk navigasjonsradar (appendiks A). Figur a) viser radarens gain i horisontalplanet til radaren, altså xz-planet i radarramme $\mathcal{F}^{Ri}$. Figur b) viser radarens gain i vertikalplanet til radaren, altså yz-planet i radarramme $\mathcal{F}^{Ri}$.



Figur 3.3: Strålingsdiagram for navigasjonsradar. a) Navigasjonsradarens gain i horisontalretningen til radaren. b) Navigasjonsradarens gain i vertikalretningen til radaren.

Vinkler mellom radarramme *i* og propagasjonsvektor

Radarens gain i retning av propagasjonsvektoren $\vec{q}_{RiSj}$ bestemmes ved vinklene θ_{Ri} og ϕ_{Ri} i radarrammen, og beregnes ved

$$\theta_{RiSj} = \text{atan2}\left(\sqrt{\left(q_{RiSj}^{Sj}(1)\right)^2 + \left(q_{RiSj}^{Sj}(2)\right)^2}, q_{RiSj}^{Sj}(3)\right) \tag{3.16}$$

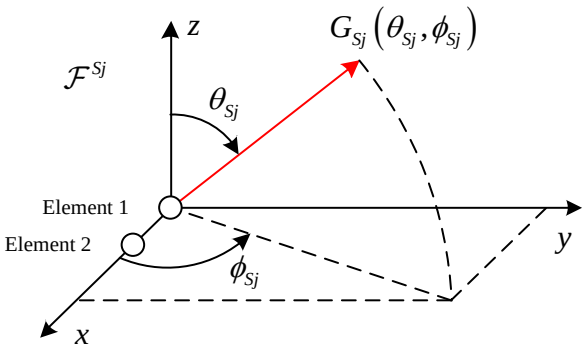
og

$$\phi_{RiSj} = \text{atan2}\left(q_{RiSj}^{Sj}(2), q_{RiSj}^{Sj}(1)\right), \tag{3.17}$$

hvor $\underline{q}_{RiSj}^{Sj} = \left[q_{RiSj}^{Sj}(1); q_{RiSj}^{Sj}(2); q_{RiSj}^{Sj}(3)\right]$. Vinklene vil være definert i intervallene $0 \leq \theta_T \leq \pi$ og $-\pi \leq \phi_T \leq \pi$.

Strålingsdiagram for dipolantenne

En dipolantenne består av to strålingskilder. I oppgaven modelleres dipolantennene ved å plassere to strålingskilder på x-aksen i sensorrammen $\mathcal{F}^{Sj}$, se Figur 3.4.

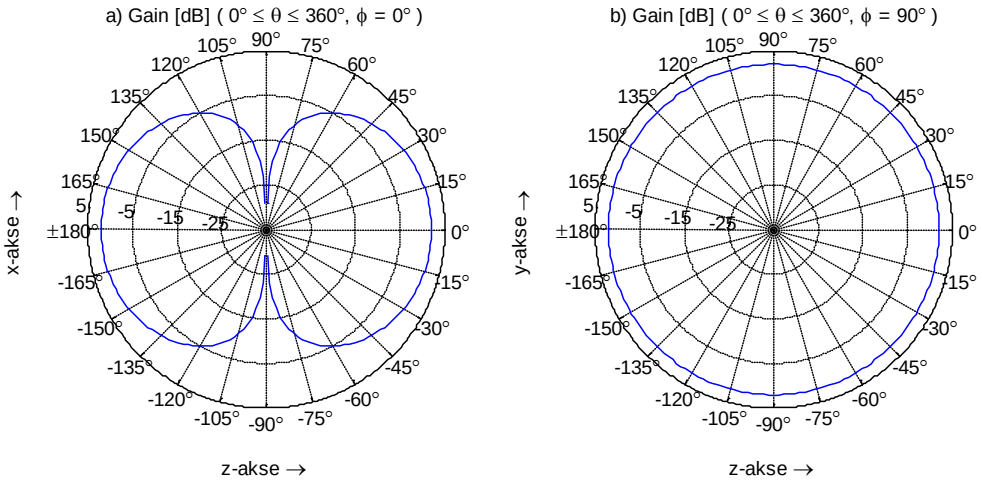


Figur 3.4: Dipolantenne med to elementer plassert utover x-aksen i sensorramme $\mathcal{F}^{Sj}$.

Dipolantennens gain beregnes ved formel (3.15). Ettersom dipolantennen ikke ligger på noe jordplan, vil elementenes forsterkning være tilnærmet omnidireksjonal, og antennens gain i retning $\left(\theta_{Sj},\phi_{Sj}\right)$ kan beregnes ved

$$G_{Sj}\left(\theta_{Sj},\phi_{Sj}\right)=\frac{1}{\sqrt{2}}\left|\sum_{n=1}^{2}\exp\left(jk_0na\sin\theta_{Sj}\cos\phi_{Sj}\right)\right|. \tag{3.18}$$

Figur 3.5 viser strålingsdiagrammet til en dipolantenne som simuleres i simulatoren. Antennen har en avstand mellom strålingskildene på en halv bølgelengde og har et gain på 2 dBi, som tilsvarer en typisk dipolantenne FFI bruker til å motta radarpulser med. Figur a) viser gain i xz-planet i sensorramme $\mathcal{F}^{Sj}$. Figur b) viser dipolannens gain i yz-planet i sensorrammen. I dette planet er gain-et konstant for alle θ-vinkler.



Figur 3.5: Strålingsdiagram for dipolantenne. a) Dipolantennens gain i xz-planet i sensorrammen $\mathcal{F}^{Sj}$. b) Dipolantennens gain i yz-planet i sensorrammen $\mathcal{F}^{Sj}$.

Vinkler mellom sensorramme *j* og propagasjonsvektor
Dipolantennens forsterkning i retning av propagasjonsvektoren $\vec{q}_{SjRi}$ bestemmes ved vinklene θ_{Sj} og ϕ_{Sj} i sensorrammen, og beregnes ved

$$\theta_{SjRi} = \text{atan2}\left(\sqrt{\left(q_{SjRi}^{Sj}(1)\right)^2 + \left(q_{SjRi}^{Sj}(2)\right)^2}, q_{SjRi}^{Sj}(3)\right) \tag{3.19}$$

og

$$\phi_{SjRi} = \text{atan2}\left(q_{SjRi}^{Sj}(2), q_{SjRi}^{Sj}(1)\right). \tag{3.20}$$

Vinklene vil være definert i intervallene $0 \leq \theta_T \leq \pi$ og $-\pi \leq \phi_T \leq \pi$.

3.1.3.2 Mottak av radarpulser

Mottatt effekt i mottaker
Forholdet mellom mottatt effekt og utsendt effekt, når sensor ligger i en avstand $\left\|\underline{q}_{RiSj}^{N}\right\|$ fra radaren, gis ved formel (2-119) i [8]

$$\frac{P_{Sj}}{P_{Ri}} = \left(\frac{\lambda}{4\pi\left\|\underline{q}_{RiSj}^{N}\right\|}\right)^2 G_{Ri}(\theta_{RiSj}, \phi_{RiSj}) G_{Sj}(\theta_{SjRi}, \phi_{SjRi}), \tag{3.21}$$

hvor P_{Sj} er mottatt effekt i sensor j, P_{Ri} er utsendt effekt fra radar i, λ er bølgelengden, $\left\|\underline{q}_{RiSj}^{N}\right\|$ er avstanden mellom radar og sensor, G_{Ri} er gain til radaren og G_{Sj} er gain til sensoren. For at sensoren skal motta radarpulser fra navigasjonsradaren, må den mottatte effekten være høyere enn en terskel. Den mottatte effekten må derfor tilfredsstille ulikheten

$$P_{Sj,min} \leq P_{Ri}\left(\frac{\lambda}{4\pi\left\|\underline{q}_{RiSj}^{N}\right\|}\right)^2 G_{Ri}(\theta_{RiSj}, \phi_{RiSj}) G_{Sj}(\theta_{SjRi}, \phi_{SjRi}). \tag{3.22}$$

I oppgaven er terskelen satt til -60 dBm, ettersom dette tilsvarer en typisk mottaker for radarpulser.

3.1.3.3 Ankomsttiden til en puls i mottaker

Radarpulsene beveger seg med en hastighet tilnærmet lik lysets hastighet. I oppgaven antas det at pulsene propagerer i den rette linjen mellom radaren og mottakeren. Tiden det tar fra en radarpuls sendes ut fra navigasjonsradaren til den når mottakeren er gitt ved

$$t = \frac{1}{c}\left\|\underline{q}_{RiSj}^{N}\right\|, \tag{3.23}$$

hvor $\left\|\underline{q}_{RiSj}^{N}\right\|$ er avstanden mellom radaren og sensoren og c er lyshastigheten. Dersom pulsen sendes ut i tidspunkt t_k, vil ankomsttiden i sensor j være

$$t_k^j = t_k + \frac{1}{c}\left\|\underline{q}_{RiSj}^N\right\|. \tag{3.24}$$

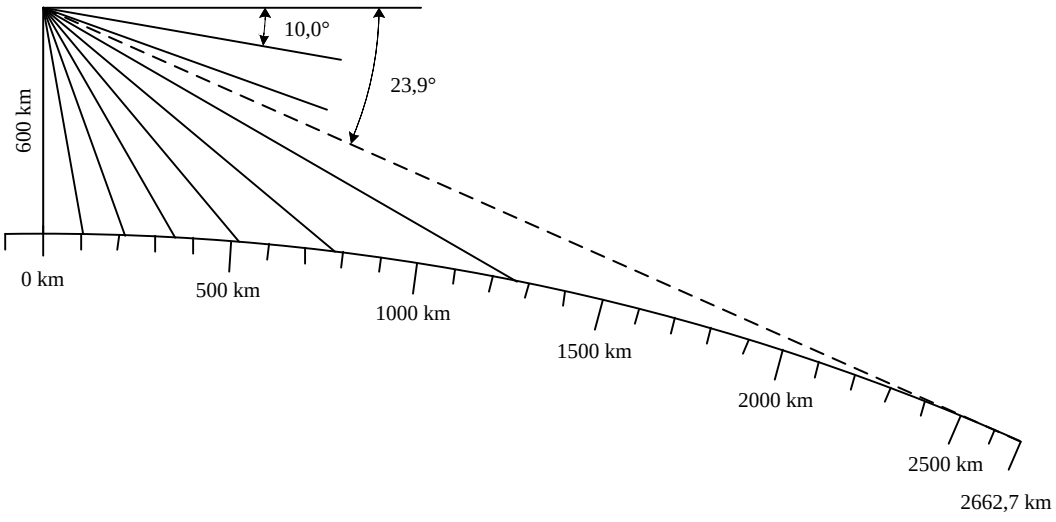
I oppgaven antas det at UAS-ene står i ro når pulsene propagerer, og at UAS-ene dermed har samme posisjon når de mottar pulsene som når pulsen ble sendt ut av radaren. Grunnen til denne antakelsen er at det tar en puls 167 µs å propagere 50 km, og med en flyhastighet på 20 m/s vil UAS-ene forflytte seg 3,33 mm. Denne ekstra forflytningen av UAS-ene kan gi opptil 11,1 ps ekstra propagasjonstid. Denne ekstra propagasjonstiden er mye mindre enn nøyaktigheten til klokka, og vil derfor bare ikke være målbar.

3.2 Matematiske modeller for satellitt-scenario

3.2.1 Jordmodell

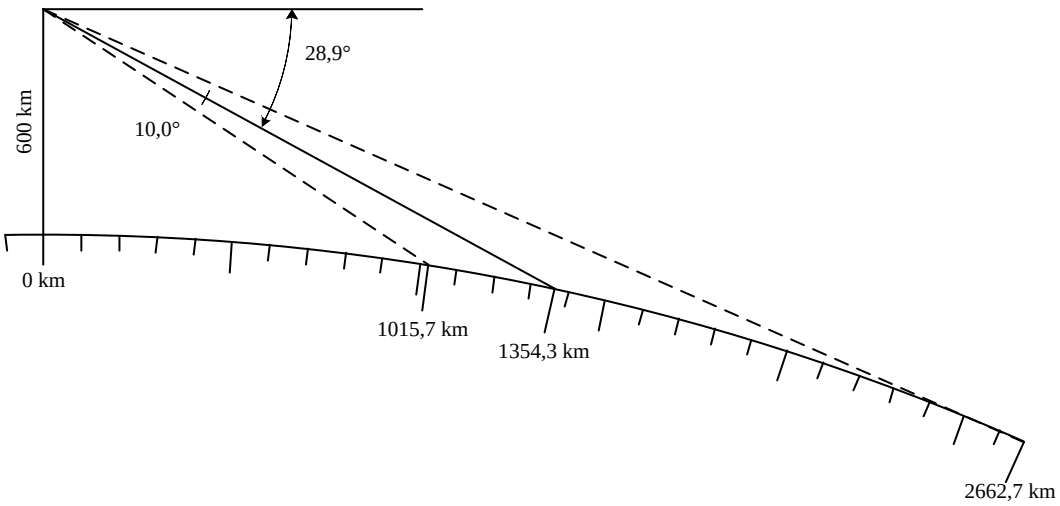
3.2.1.1 Kuleformet jord

De to satellittene vil ligge i en lav jordbane, 600 km over jordoverflaten. Figur 3.6 viser en satellitt som ligger i en høyde på 600 km over jordoverflaten på en kuleformet jord. Figuren viser at elevasjonsvinkelen mot horisonten er 23,9°. Avstanden ut til horisonten, nede jo jordoverflaten, fra satellittbanen er 2662,7 km. I oppgaven antas det at pulsene vil propagere i den rette linjen mellom radarene og satellittene, og det vil derfor ikke være mulig å motta pulser fra radarer som ligger bak jordkrumningen. I figuren er det tegnet opp ulike hovedlober med lobebredde på 10°. Som figuren viser blir det opplyste området, altså den delen av jordoverflaten som satellittens hovedlobe dekker, større når hovedloben peker ut mot horisonten.



Figur 3.6: Satellitt i en lav jordbane 600 km over en kuleformet jord [9].

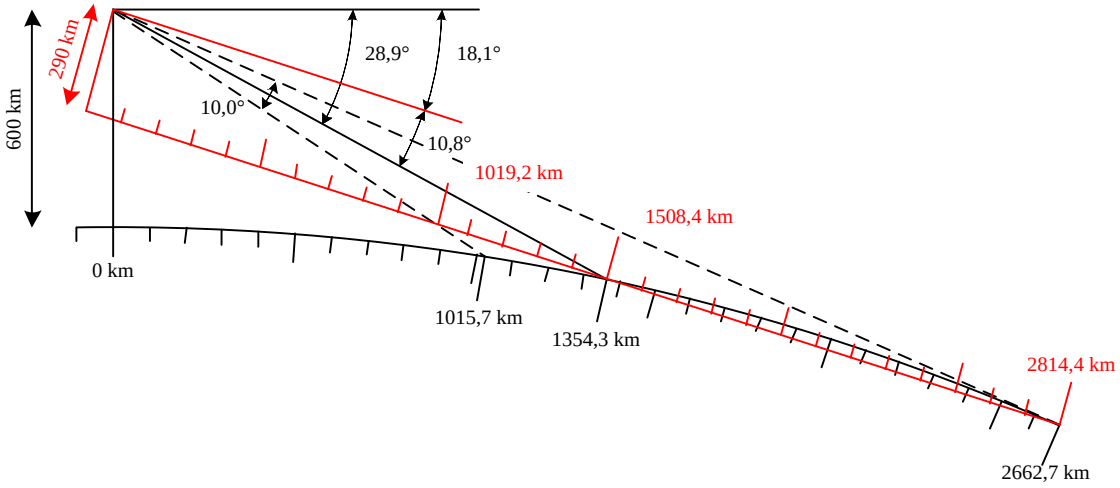
Dersom det ønskes å geolokalisere skip i et størst mulige område, vil det være naturlig at hovedloben peker ut mot horisonten. Figur 3.7 viser hovedloben til satellitten dersom den peker 28,9° ned. Det opplyste området vil da strekke seg fra 1015,7 km og ut til horisonten på 2662,7 km.



Figur 3.7: Hovedloben til en satellitt i en lav jordbane 600 km over en kuleformet jord.

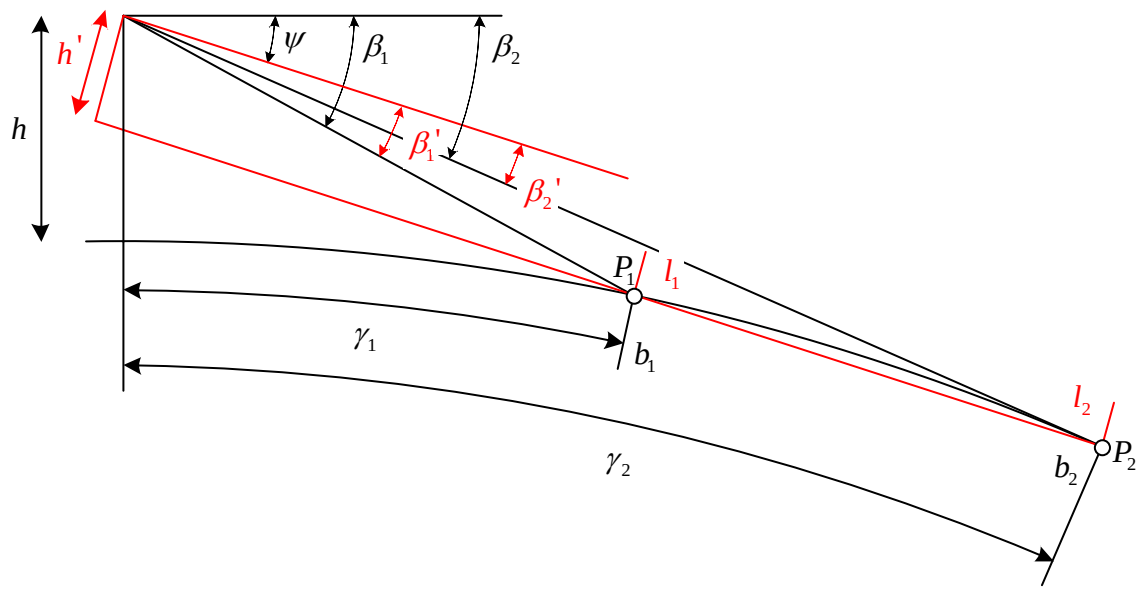
3.2.1.2 Flat jord

For at avstandene i det opplyste området skal være mest mulig lik for kuleformet jord og flat jord, velges det at den flate jorden skjærer den kuleformede jorden midt i hovedloben og i horisonten. Figur 3.8 viser sammenhengen mellom kuleformet jord og den flate jorden. I dette tilfellet vil det opplyste området ligge mellom 1019,2 km og 2814,4 km for flat jord. For den kuleformede jorden beveger satellittene seg med konstant høyde jordoverflaten. Dette fører til at satellittene vil fly med konstant høyde over et plan, når det antas at jorden er flat.



Figur 3.8: Kuleformet jord (sort) og flat jord (rødt).

Satellittenes høyde og pekeretning vil være ulik for flat jord kontra kuleformet jord. Figur 3.9 definerer ulike vikler som benyttes for å gå fra kuleformet jord til flat jord.



Figur 3.9: Definisjon av vinkler ved overgang fra kuleformet jord til flat jord.

Den flate jorden bestemmes ved at den skjærer den kuleformede jorden i to punkter, P_1 og P_2, bestemt ved elevasjonsvinklene β_1 og β_2. For å beregne avstanden ut til punktene, P_1 og P_2, nede på jordoverflaten brukes definisjonen for buelengde. I formelen for buelengde inngår vinkelen mellom satellitten og punktet, samt jordradiusen. Vinkelen mellom satellitten og punktet bestemmes ved

$$\gamma_i = \beta_i + \arcsin\left(\left(\frac{r+h}{r}\right)\cos(\beta_i)\right) - \frac{\pi}{2}, \qquad i = 1,2, \tag{3.25}$$

hvor r er radius til jorden, og h er satellittens høyde over jordoverflaten. Avstanden fra satellitt ut til punktet nede på jordoverflaten (altså buelengden) bestemmes ved

$$b_i = r \cdot \gamma_i, \qquad i = 1,2. \tag{3.26}$$

Elevasjonsvinklene for den flate jorden og kuleformede jorden er ulike. For å bestemme elevasjonsvinklene for den flate jorden, trengs hjelpevinkelen

$$\psi = \frac{1}{2}(\gamma_1 + \gamma_2). \tag{3.27}$$

Elevasjonsvinkelen i det nye jordplanet blir da

$$\beta_i' = \beta_i - \psi, \qquad i = 1,2. \tag{3.28}$$

Høyden til satellittene i det nye jordplanet blir

$$h' = \frac{r \sin\gamma_i \sin\beta_i'}{\cos\beta_i}. \tag{3.29}$$

Avstanden ut til punktet P_i i det nye jordplanet bestemmes ved

$$l_i = \frac{h'}{\tan \beta_i}. \tag{3.30}$$

3.2.1.3 Banehastighet for flat jord

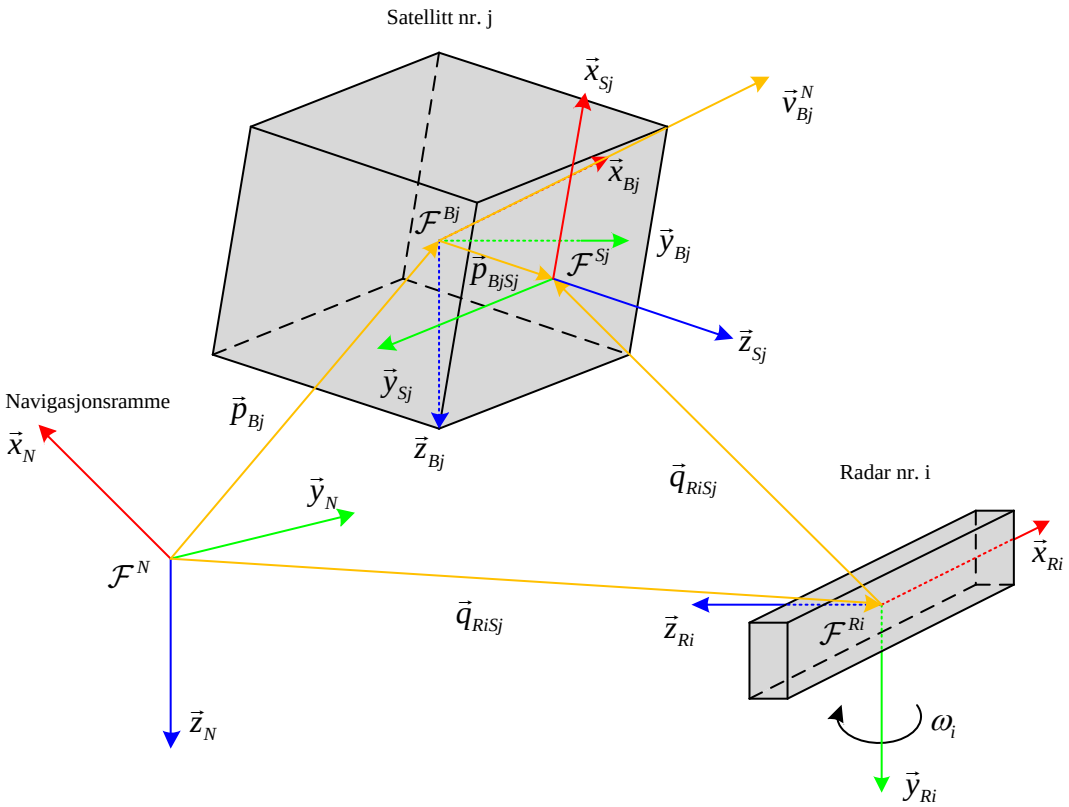
For en sirkelformet satellittbane vil hastigheten til satellittene være ulik i satellittbanen og nede på jordoverflaten. For flat jord er hastigheten til satellittene lik i banen og nede på jordoverflaten. Hastigheten til satellitten for flat jord vil derfor være lik hastigheten til satellittene nede på jordoverflaten for kuleformet jord, og vil være gitt ved

$$v_1 = \frac{r}{r+h} v_0 \approx 6854\,\mathrm{m/s}, \tag{3.31}$$

hvor r er den kuleformede jordens radius, h høyden til satellittene, og v_0 den tangentielle hastigheten i satellittbanen.

3.2.2 Definisjon av rammer og vektorer

For å beskrive posisjon og orientering til sensorplattformene og navigasjonsradarene, trengs det et sett med rammer og vektorer mellom dem. Figur 3.10 viser en oversikt over rammer for å beskrive en satellitt og en navigasjonsradar, samt vektorene mellom rammene.



Figur 3.10: Rammer og vektorer for satellitt-scenario.

3.2.2.1 Rammer

Som for UAS-scenarioet innføres det en navigasjonsramme. Navigasjonsrammen er definert $\mathcal{F}^N = \{O_N; \vec{x}_N, \vec{y}_N, \vec{z}_N\}$, hvor x-aksen og y-aksen ligger i det flate jordplanet, mens z-aksen peker ned. Ettersom simuleringene kun foregår på et lite område på jorden, peker x-aksen alltid mot nord og y-aksen mot øst. Enhetene langs de tre aksene er meter.

Satellittene vil kunne roteres uavhengig av satellittbanen. Innfører derfor en baneramme som ligger fast i satellittbanen og en sensorramme som beskriver orienteringen til antennen. Banerammen bestemmes ved $\mathcal{F}^{Bj} = \{O_{Bj}; \vec{x}_{Bj}, \vec{y}_{Bj}, \vec{z}_{Bj}\}$ for $j = 1{,}2$. Banerammen har origo i massesenteret til satellittene; og er orientert slik at x-aksen alltid peker parallelt med hastighets-vektoren, y-aksen ut mot siden og z-aksen ned. Det betyr at banerammen for satellitt-scenarioet er definert på samme måte som bodyrammen i UAS-scenarioet, og de samme formlene kan benyttes for begge scenarioene. Orienteringen til satellittene vil være bestemt ved sensorrammen $\mathcal{F}^{Sj} = \{O_{Sj}; \vec{x}_{Sj}, \vec{y}_{Sj}, \vec{z}_{Sj}\}$ for $j = 1{,}2$. Sensorrammen har origo i senter av antennen; og er orientert slik at z-aksen alltid peker rett frem i hovedloben. x- og y-aksen er plassert slik at rammen danner et høyrehåndssystem.

Radarrammene i satellitt-scenarioet defineres på samme måte som for UAS-scenarioet i avsnitt 3.1.1.1.

3.2.2.2 Vektorer

Posisjonsvektorer, propagasjonsvektorer og hastighetsvektorer er definert på samme måte som for scenario med to UAS-er, se avsnitt 3.1.1.

3.2.3 Beregning av vektorer og koordinattransformasjonsmatriser

I likhet med UAS-scenarioet, oppdateres satellittenes posisjon og orientering, og radarens orientering kun i pulsutsendelsestidspunktene. Det forutsettes da at satellittenes hastigheter og rotasjonshastigheter samt navigasjonsradarenes rotasjonshastigheter er konstant mellom pulsenes utsendelsestidspunkter.

Rammene og vektorene for satellitt-scenarioet, bortsett fra sensorrammen, er definert på samme måte som i UAS-scenarioet. Beregninger av KTM, posisjonsvektorer og propagasjonsvektorer er derfor identisk beregningene i avsnitt 3.1.2, bortsett fra beregningen av KTM fra sensorramme til banerammen. Denne blir ulik ettersom satellittene vil kunne roteres uavhengig av bevegelsesret-ningen.

3.2.3.1 Koordinattransformasjonsmatriser

Koordinattransformasjonsmatrise fra sensorramme til baneramme på satellitt *j*
Pekeretningen til antennen og dermed orienteringen til satellittene vil være avhengig av hvilket område det ønskes at hovedlobene skal dekke. Sensorrammen er definert slik at z-aksen alltid peker rett frem i hovedloben til antennen. Lar orienteringen til sensorrammen defineres av en asimutvinkel og en elevasjonsvinkel. Asimutvinkelen vil være vinkelen mellom x-aksen til

banerammen og z-aksen til sensorrammen i xy-planet til banerammen. Asimutvinkelen vil da være gitt ved

$$\alpha = \text{atan2}\left(z_{Sj}^{Bj}(2), z_{Sj}^{Bj}(1)\right). \tag{3.32}$$

Elevasjonsvinkelen vil være vinkelen mellom xy -planet til banerammen og z-aksen til sensorrammen. Elevasjonsvinkelen vil da være gitt ved

$$\beta = \text{atan2}\left(z_{Sj}^{Bj}(3), \sqrt{\left(z_{Sj}^{Bj}(1)\right)^2 + \left(z_{Sj}^{Bj}(2)\right)^2}\right). \tag{3.33}$$

Sensorrammen vil da være lik banerammen, rotert en vinkel α om z-aksen i banerammen, og deretter en vinkel $\frac{\pi}{2} - \beta$ og den nye y-aksen. KTM fra sensorrammen til banerammen blir da

$$R_{Sj}^{Bj} = \left[\boldsymbol{R}_{BjSj}\right]^{Bj} = R_z(\alpha)R_y\left(\frac{\pi}{2} - \beta\right). \tag{3.34}$$

3.2.4 Propagasjon av radarpulser

Ettersom det ikke er utført noen forsøk på å motta radarpulser fra satellitter, antas det at satellittene vil motta pulser som sendes ut fra en radars hovedlobe og som mottas av en satellitts hovedlobe. For satellittene-scenarioet vil pulsenes propagasjonstid være så lang at satellittene vil ha beveget seg forholdsvis langt innen en puls har nådd frem. Dersom en radar ligger i en avstand på 2000 km fra satellittbanen, vil en puls bruke 6,7 ms på å propagere fra radaren og frem til satellitten. I løpet av denne tiden vil satellittene forflytte seg 46,2 m. Dette kan resultere en ekstra propagasjonstid på 154 ns. Det må derfor tas i betraktning at satellittene har forflyttet seg innen at pulsen har nådd frem til satellitten.

Ankomsttidene i sensorene beregnes da ved at sensorenes posisjoner og orienteringer samt radarens orientering beregnes i pulsens utsendelsestidspunkt. Deretter beregnes det hvor lang tid pulsen bruker på å propagere, og satellittene forflyttes og roteres deretter. Dersom satellittene kan motta pulsen etter at satellittene er forflyttet og rotert, beregnes ankomsttiden.

3.2.4.1 Mottak av radarpulser

I simuleringen vil satellittene motta pulser som sendes ut fra radarens hovedlobe og som mottas i hovedloben til satellittene, dvs. at propagasjonsvektoren tilfredsstiller

$$\left|\text{atan2}\left(\sqrt{\left(q_{SjRi}^{Sj}(1)\right)^2 + \left(q_{SjRi}^{Sj}(2)\right)^2}, q_{SjRi}^{Sj}(3)\right)\right| \leq \frac{\theta_{Sj}}{2} \tag{3.35}$$

og

$$\left|\text{atan2}\left(q_{RiSj}^{Ri}(1), q_{RiSj}^{Ri}(3)\right)\right| \leq \frac{\theta_{Ri}}{2}, \tag{3.36}$$

hvor θ_{Sj} er lobebredden til sensor j, θ_{Ri} er lobebredden til radar i, og $\underline{q}_{SjRi}^{Sj}$ er propagasjonsvektoren fra satellitten til radaren og $\underline{q}_{RiSj}^{Ri}$ fra radaren til satellitten.

3.2.4.2 Ankomsttiden til en puls i mottakeren

Radarpulsene beveger seg med en hastighet tilnærmet lik lysets hastighet, i den rette linjen mellom radar og mottaker. Tiden det tar fra en radarpuls sendes ut fra navigasjonsradaren til den når mottakeren er gitt ved

$$t = \frac{\left\| \underline{q}_{RiSj}^{N} \right\|}{c} \tag{3.37}$$

hvor $\left\| \underline{q}_{RiSj}^{N} \right\|$ er avstanden mellom radaren og sensoren og c er lyshastigheten. Dersom pulsen sendes ut i tidspunkt t_k, vil ankomsttiden i sensor j være

$$t_k^j = t_k + \frac{1}{c} \left\| \underline{q}_{RiSj}^{N} \right\|. \tag{3.38}$$

4 Pulssimulator

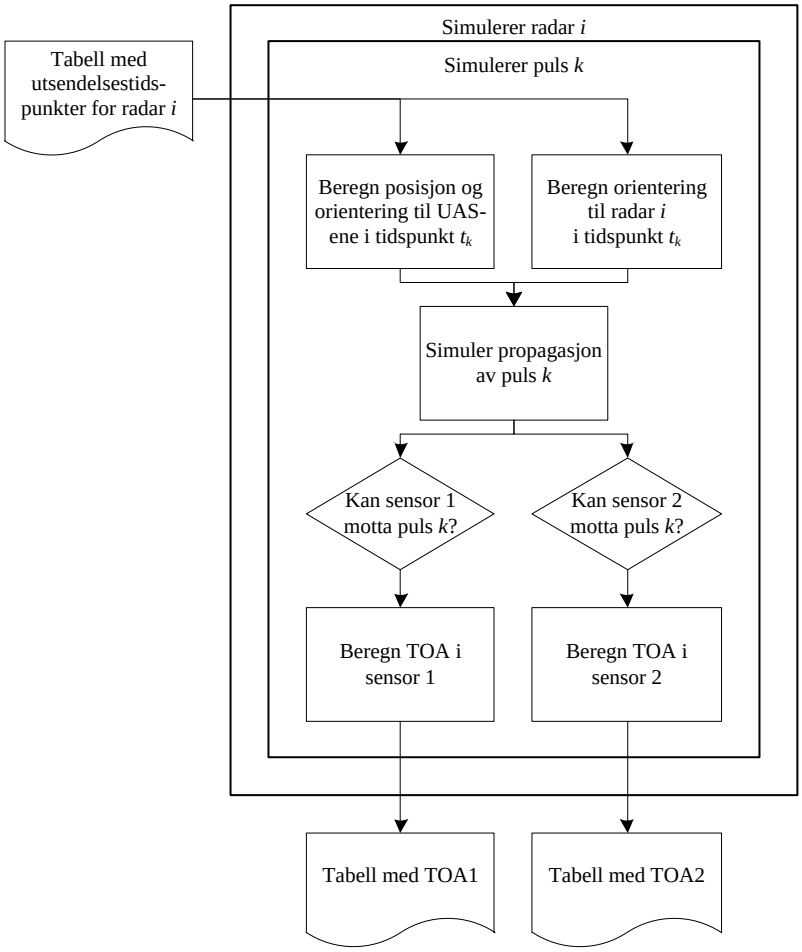
Til å generere ankomsttider til radarpulser i de to sensorplattformene, benyttes det en simulator. I simulatoren simuleres et antall roterende navigasjonsradarer og to sensorplattformer i bevegelse. Simulatoren fungerer ved at en og en radar simuleres av gangen. Hver radar bestemmes ut fra en posisjon, rotasjonshastighet, initiell pekeretning og utsendelsestidspunkter for pulsene. Utsendelsestidspunktene ligger i en tabell og består av reelle målinger av pulsers ankomsttid i en sensor. Simulatoren plukker kronologisk ut et utsendelsestidspunkt, beregner orientering til radaren, og posisjonen og orienteringen til sensorene. Dersom en sensor kan motta pulsen, beregnes ankomsttiden. Når alle pulsene fra en radar er simulert, simuleres pulser fra neste radar. Til slutt sorteres alle ankomsttidene til pulsene kronologisk, og lagres som et måleopptak.

Kapittelet er delt inn i fire deler. I avsnitt 4.1 beskrives simulatorens virkemåte for UAS-scenarioet, og i avsnitt 4.2 beskrives simulatorens virkemåte for satellitt-scenarioet. Avsnitt 4.3 inneholder en beskrivelse av pulsutsendelsestidspunktene. Avsnitt 4.4 inneholder en analyse av maksimumsavstanden mellom sensorene i simulatoren for at begge sensorene skal motta den samme pulsen.

4.1 Pulssimulator for UAS-scenarioet

I UAS-scenarioet er avstandene mellom sensorene og radarene forholdsvis små. Det betyr at sensorene vil motta pulser fra radarenes hovedlober og noen sidelober. I simulatoren simuleres derfor strålingsdiagram for dipolantennene og navigasjonsradarene. En puls vil mottas i sensorene dersom den mottatte effekten til en puls ligger over en terskel. I simulatoren benyttes realistiske verdier for utsendt effekt til pulsene, rotasjonstid til radarene og parametere i strålingsdiagrammet til dipolantennene og radarene. Appendiks H.1 inneholder pseudokode til pulssimulatoren for UAS-scenarioet, og er i henhold til den matematiske modellen beskrevet i avsnitt 3.1.

Figur 4.1 inneholder et overordnet flytskjema over simulatoren for UAS-scenarioet. Ytterste løkke simulerer radar i og innerste løkke puls k. For hvert pulsutsendelsestidspunkt oppdateres sensorplattformenes posisjon og orientering, og radarens orientering. Dersom pulsen kan mottas i en av sensorene, beregnes ankomsttiden til pulsen.

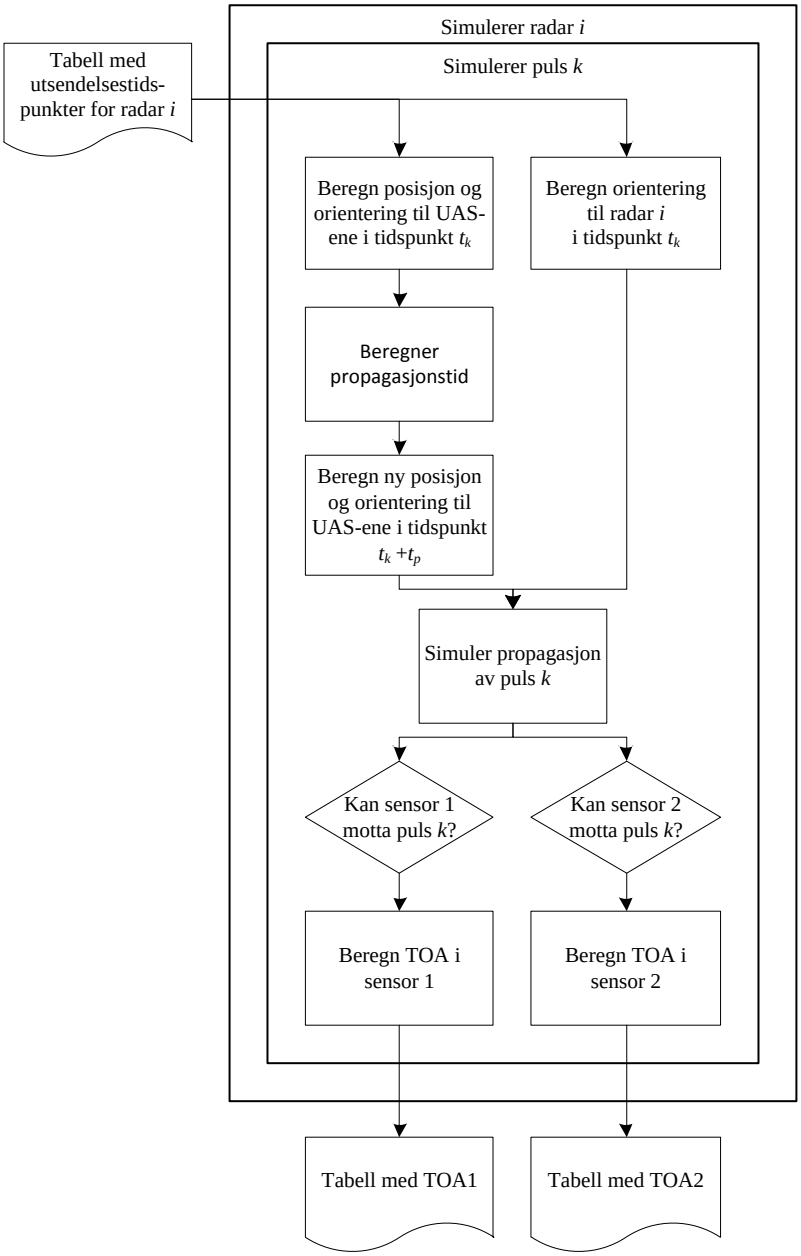


Figur 4.1: Flytskjema for simulering av radarpulser i UAS-scenarioet.

4.2 Pulssimulator for satellitt-scenarioet

I analyser utført på FFI ser det ut til at to satellitter i en lav polarbane vil motta radarpulser fra navigasjonsradarer. Sannsynligvis vi satellittene kun motta pulser fra radarenes hovedlober, og som mottas innenfor hovedlobene til de retningsbestemte antennene på satellittene. I simulatoren for satellitt-scenarioet antas det derfor at satellittene mottar alle pulser som sendes fra en radars hovedlobe og som mottas innenfor en satellitts hovedlobe. I simulatoren vil satellittene forflytte seg under propagasjonstiden, og satellittenes posisjon og orientering må derfor beregnes i tidspunktet når pulsen mottas i satellittene. Appendiks H.2 inneholder pseudokode til pulssimulatoren for satellitt-scenarioet, og er i henhold til den matematiske modellen beskrevet avsnitt 3.2.

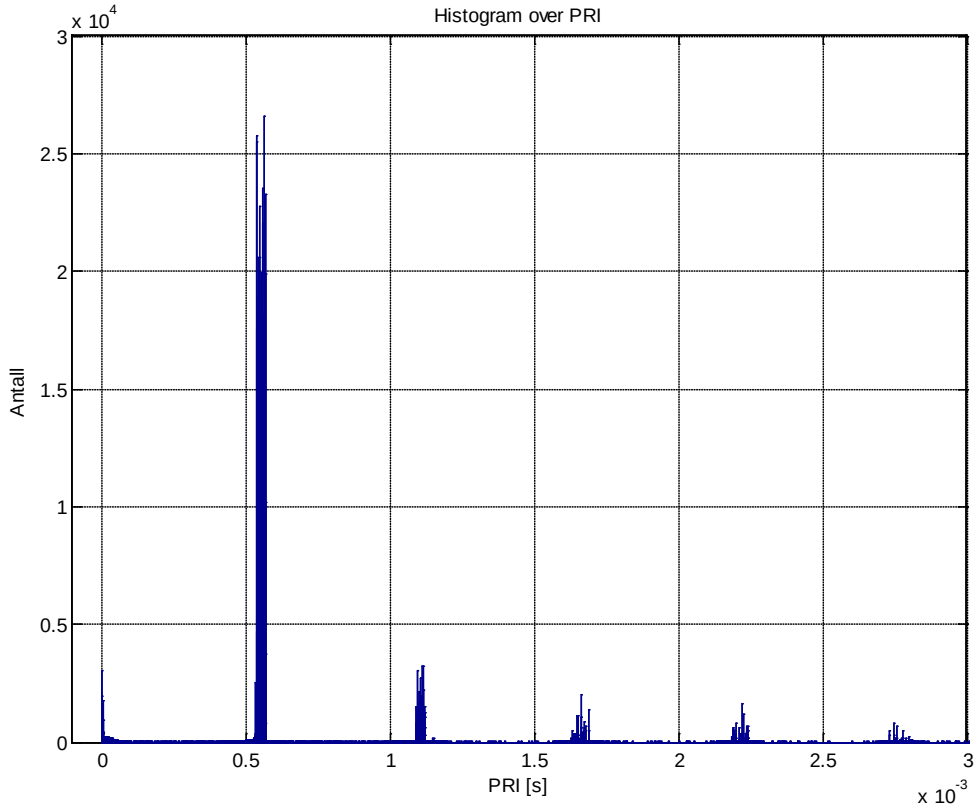
Figur 4.2 inneholder et overordnet flytskjema over simulatoren for satellitt-scenarioet. Ytterste løkke simulerer radar i og innerste løkke puls k. For hvert pulsutsendelsestidspunkt oppdateres sensorplattformenes posisjon og orientering, og radarens orientering. Dersom pulsen kan mottas i en av sensorene, beregnes ankomsttiden til pulsen.



Figur 4.2: Flytskjema for simulering av radarpulser i satellitt-scenarioet.

4.3 Pulssutsendelsestidspunkter

For at ankomsttidene i de to sensorene skal likne mest mulig på reelle ankomsttider i to sensorer, benyttes pulsutsendelsestidspunkter fra reelle målinger. Pulsutsendelsestidspunktene er basert på opptak av ankomsttider til radarpulser fra en og en radar. Opptakene inneholder kun ankomsttider fra hovedlobepasseringer til radarene, og er derfor skjøtet sammen til en kontinuerlig strøm av ankomsttider [10]. Figur 4.3 et histogram over PRI til radarpulsene for en av radarene i simulatoren. På grunn av PRI-stagger (se appendiks A) hopper radaren mellom blant annet frekvensene 1800 Hz (PRI lik 0,56 ms), 900 Hz (PRI lik 1,11 ms), og 600 Hz (PRI lik 1,67 ms). Det ser også ut til at radaren har PRI-jitter ved at hver stagger består av ulike PRI-er (flere stolper ved siden av hverandre). PRI-ene nært 0 s (rundt 0,5 µs) er feil i datasettet som ikke har blitt fjernet.



Figur 4.3: Histogram over PRI til pulsene fra en navigasjonsradar i simulatoren. Histogrammet inneholder PRI fra 568604 radarpulser i et 423 s langt opptak. Bredden på stolpene er 1 µs.

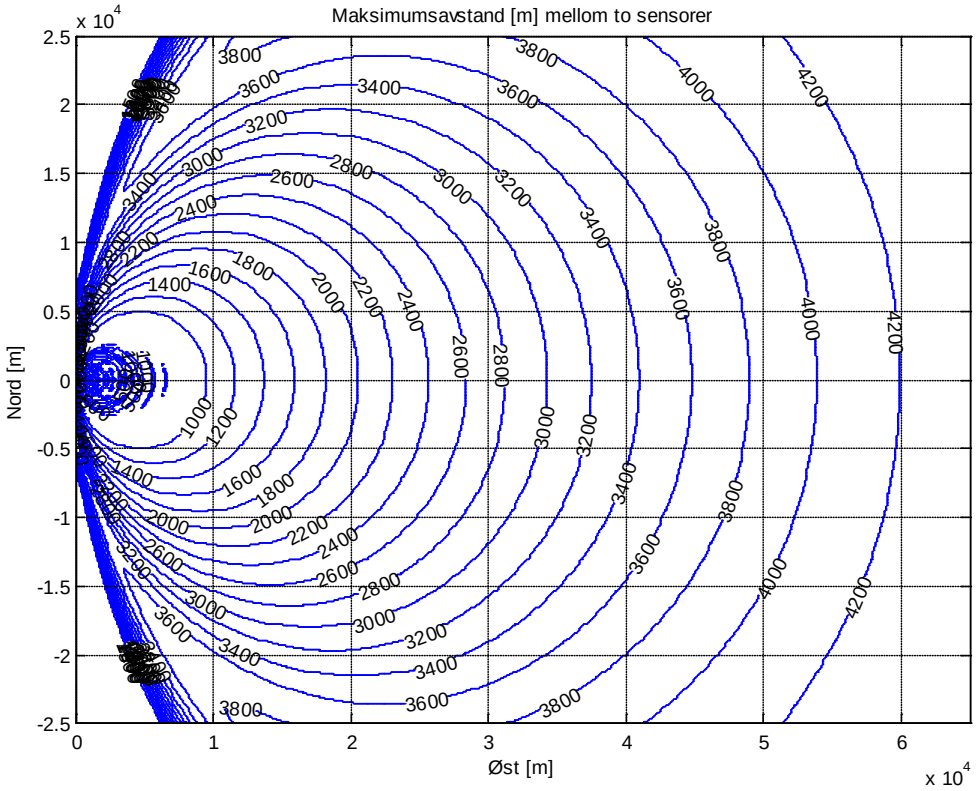
4.4 Maksimumsavstand mellom sensorplattformene i simulatoren

For at radarer skal kunne geolokaliseres vha. TDOA, må begge sensorplattformene motta den samme pulsen. Avstanden mellom sensorene påvirker både minimumsavstanden ut til radarene og estimeringsnøyaktigheten. Når avstanden mellom sensorene øker; blir estimeringsnøyaktigheten bedre, men samtidig må radarene ligge lengre unna for at sensorene skal kunne motta pulsene. Tilsvarende gjelder når avstanden mellom sensorene minker; da blir estimeringsnøyaktigheten dårligere, men avstanden ut til radarene kan også være mindre. Ettersom det er avgjørende at begge sensorene mottar de samme pulsene, bestemmes avstanden mellom sensorene primært av at begge sensorene må motta de samme pulsene.

I avsnitt 4.4.1 ses det på maksimumsavstanden mellom UAS-ene for at sensorene skal motta de samme pulsene fra radarer på en viss avstand, og i avsnitt 4.4.2 ses det på maksimumsavstanden mellom satellittene.

4.4.1 Maksimumsavstand mellom UAS-ene

Ettersom avstanden mellom UAS-ene og navigasjonsradarene er små, må det forventes at sensorene mottar pulser fra radarenes sidelober og baklober. Ved å benytte formlene for mottak av radarpulser fra pulssimulatoren, kan ulike avstander mellom UAS-ene testes ut. Figur 4.4 viser et konturplott over hva maksimumsavstanden mellom sensorene må være for at begge sensorene skal motta de samme pulsene. I figuren er sensorene plassert symmetrisk om origo, langs y-aksen, og radaren peker mot origo. Til å bestemme avstanden mellom sensorene, er det benyttet formler for propagasjon av radarpulser fra avsnitt 3.1.3. Dvs. at det er benyttet strålingsdiagram til navigasjonsradaren og dipolantennene på UAS-ene, utsendereffekt og minimumseffekt for mottak av pulser i mottaker. Figuren viser at avstanden mellom sensorene må være størst når radarer ligger på x-aksen i figuren, mens avstanden kan være mindre når radarenes posisjon avviker fra denne linjen. Dette skyldes av vinkelen mellom sensorene, sett fra radaren, er størst når radaren står normalt på linjen mellom sensorene.

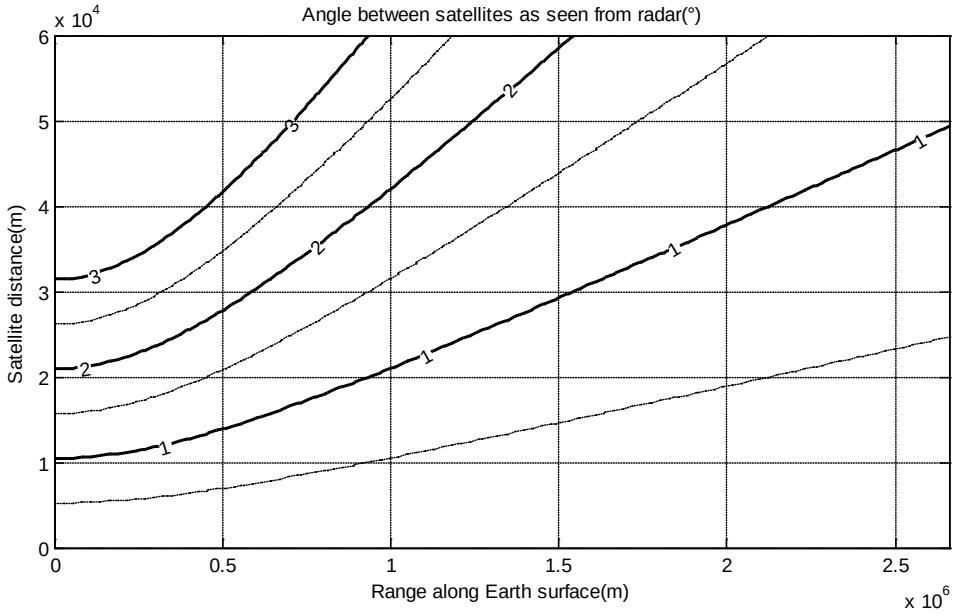


Figur 4.4: Maksimumsavstand mellom to sensorer for at de begge skal motta den samme pulsen.

I avsnitt 2.1 ble scenarioet for to UAS-er beskrevet, og det ble det skrevet at avstandene ut til radarene ville ligge i området 10-50 km. I oppgaven benyttes det en avstand på 1500 m mellom sensorene, og minimumsavstanden ul til radarene som skal geolokaliseres blir rundt 15 km. Radarer som ligger nærmere vil trolig ikke bli geolokalisering.

4.4.2 Maksimumsavstand mellom satellitter

Som for scenario med to UAS-er begrenses avstanden mellom satellittene ved at begge satellittene må kunne motta den samme pulsen som sendes ut fra en radar. Figur 4.5 viser maksimumsavstanden mellom satellittene for en gitt avstand ut til radaren nede på jordoverflaten. Hver kurve i figuren indikerer ulike vinkler mellom satellittene sett fra en radar. y-aksen indikerer ulike avstander mellom satellittene, og x-aksen avstanden fra satellittbanen og ut til radaren. I figuren antas det kuleformet jord.



Figur 4.5: Vinkel mellom to satellitter, sett fra en navigasjonsradar [3].

I oppgaven peker hovedloben ut mot horisonten, og dekker området fra 1015,7 km og ut til horisonten på 2662,7km. En navigasjonsradar har typisk en lobebredde 1-6° (se appendiks A). Ved å velge at avstanden mellom satellittene er 30 km, vil satellittene motta pulser fra radarer som har en hovedlobebredde som er større 1,5° (i oppgaven benyttes det radarer med lobebredde på 3°).

5 Beregning av TDOA og målemodell

En tidsdifferanse (TDOA) er tiden det tar fra en puls mottas i en sensor til den mottas i en annen sensor. Dersom flere sensorer mottar pulser på ulike steder, kan radaren som sendte ut pulsene lokaliseres ved å bruke TDOA. Beregning av TDOA fra to UAS-er er godt beskrevet i artiklene [5] og [6], men i begge artiklene mottas det bare pulser fra en radar.

Fra kapittel 3 fås matematiske modeller for sensorplattformene og radarer. I dette kapittelet benyttes det kun sensorenes posisjoner ved pulsenes ankomsttidspunkt, og det skilles ikke mellom de to scenarioene. Radarenes posisjoner vil være ukjent og benevnes derfor $\underline{x}^N$.

I avsnitt 5.1 mottar to sensorer en og en puls fra en radar, og det er ingen tvil om hvilke pulser som hører sammen i de to sensorene. Avsnittet bygger på formler fra artiklene [5] og [6] . I avsnitt 5.2 mottar to sensorer pulser fra flere radarer. Det vil da være usikkert hvilke pulser som hører sammen i de to sensorene. Avsnitt 5.3 inneholder en beskrivelse av støyen som er på ankomsttidene i de to sensorene og hva denne skylles.

5.1 TDOA mellom en og en radarpuls

5.1.1 Deterministiske målinger

En radar i sender ut en puls i tidspunkt t_k fra posisjonen $\underline{x}^N = \left[x_{Ri}^N; y_{Ri}^N; z_{Ri}^N\right]$. Pulsen mottas av en sensor j i posisjonen $\underline{p}_{Sj}^N(k) = \left[x_{Sj}^N; y_{Sj}^N; z_{Sj}^N\right]$. Ankomsttiden til pulsen i sensor j, t_k^j, er gitt ved [6]

$$t_k^j = t_k + \frac{r_k^{ij}}{c}, \tag{5.1}$$

hvor c er lysets hastighet og r_k^{ij} er avstanden mellom radar i og sensor j, og er gitt ved

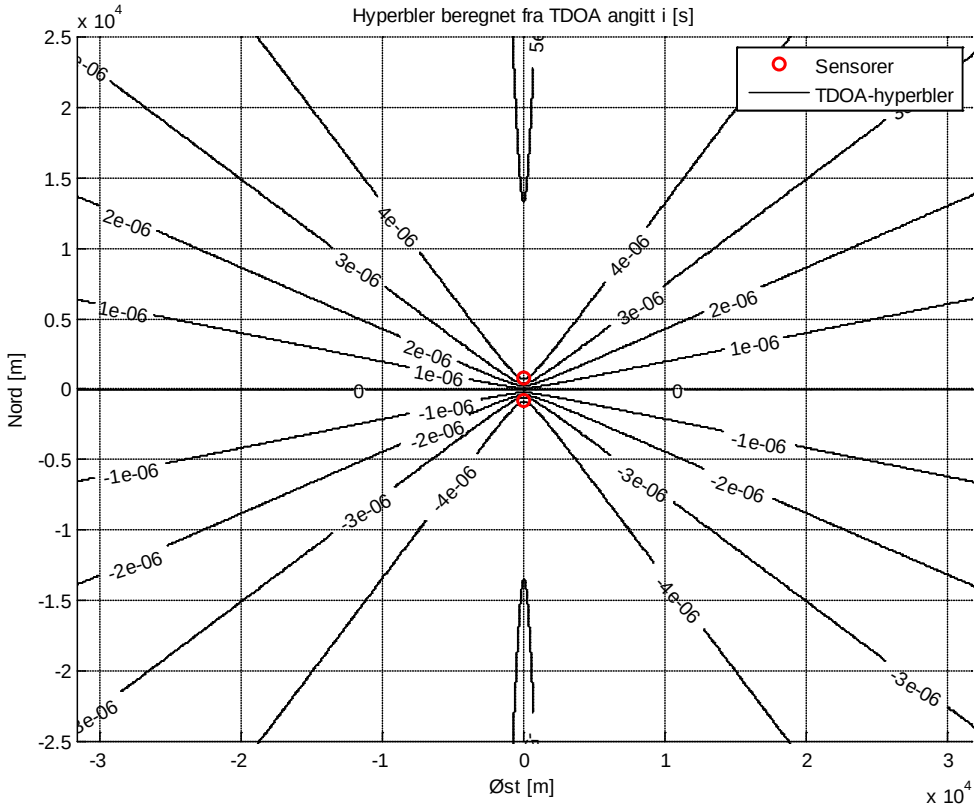
$$r_k^{ij} = \left\| \underline{x}^N - \underline{p}_{Sj}^N(k) \right\| = \sqrt{\left(x_{Ri}^N - x_{Sj}^N\right)^2 + \left(y_{Ri}^N - y_{Sj}^N\right)^2 + \left(z_{Ri}^N - z_{Sj}^N\right)^2}. \tag{5.2}$$

Ved å beregne TDOA mellom ankomsttiden til pulsene, elimineres utsendelsestidspunktet. TDOA mellom sensorene er gitt ved [6]

$$z_k = t_k^2 - t_k^1 = \frac{1}{c}\left(r_k^{i2} - r_k^{i1}\right). \tag{5.3}$$

Ettersom TDOA beregnes av kjente ankomsttider og at sensorenes posisjoner i ankomsttidene er kjent, består formel (5.3) av tre ukjente. Radaren som sendte pulsen må da ligge på en hyperboloide. Ved å anta at radaren ligger på jordoverflaten, reduseres formel (5.3) til to ukjente, og radaren må ligger i skjæringa mellom hyperboloiden og jordoverflaten, altså på en hyperbel (for med informasjon om hyperboloider og hyperbler, se appendiks C). TDOA vil alltid ligge i intervallet $[-d/c\,, d/c]$, hvor d er avstanden mellom sensorene, ettersom TDOA maksimalt kan være den tiden det tar for en puls å propagere i den rette linjen mellom sensorene. Formel (5.3) har derfor ingen reelle løsninger for TDOA utenfor intervallet $[-d/c\,, d/c]$. Figur 5.1 viser

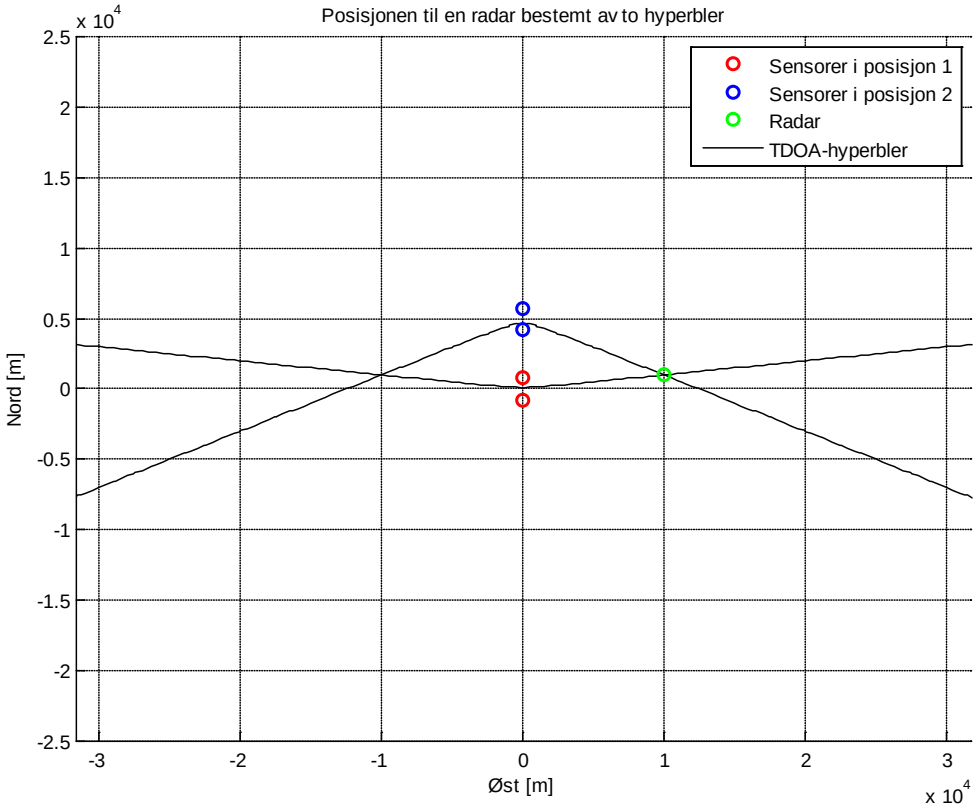
hyperbler for noen utvalgte TDOA-er. For en gitt TDOA kan radaren ligge i ethvert punkt på hyperbelen. Hyperblene vil alltid være symmetrisk om aksen som går gjennom sensorene.



Figur 5.1: Hyperbler på jordoverflaten for utvalgte TDOA. Sensorene har en innbyrdes avstand på 1500 m og en høyde på 500 m.

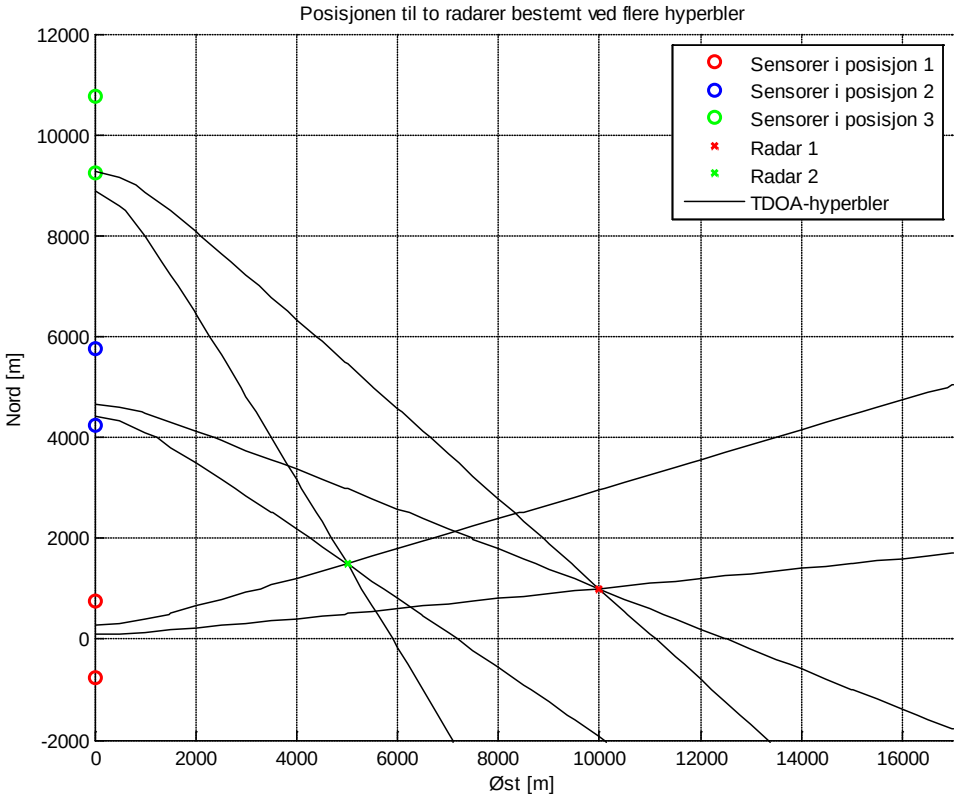
For å bestemme posisjonen til en radar trengs det flere TDOA-er. Tre sensorer vil gi tre TDOA, og dermed tre hyperbler. Radarposisjonen bestemmes da ved skjæringspunktet mellom de tre hyperblene. Ettersom hyperblene er symmetrisk om aksen gjennom sensorene, må symmetrien «brytes» for å oppnå en entydig posisjon. Dette kan gjøres ved at de tre sensorene ikke ligger på en rett linje.

Alternativt kan posisjonen til radaren bestemmes ved to sensorer i bevegelse. Over tid vil sensorene motta pulser, og dermed TDOA, på ulike steder. Fra hver av TDOA-ene kan hyperbler beregnes, og radarens posisjon kan bestemmes. Figur 5.2 viser to sensorer som først mottar en puls i en posisjon og deretter flyttes for å motta en ny puls. Av dette kan det beregnes to TDOA-er, og dermed to hyperbler. I figuren skjærer hyperblene hverandre i to steder, noe som gir to mulige radarposisjoner. Sensorene må forflytte seg langs andre trajektorier enn en rett linje for at radarposisjonen skal bli entydig.



Figur 5.2: Posisjonen til en radar bestemt ved to hyperbler. Sensorene har en innbyrdes avstand på 1500 m og en høyde på 500 m.

Når to radarer skal geolokaliseres, må sensorene motta pulser fra tre ulike steder for at det skal være mulig å skille radarene fra hverandre. Figur 5.3 viser to radarer og to sensorer i bevegelse ved tre tidspunkter. Ved hvert tidspunkt mottar sensorene en puls fra hver av radarene som det beregnes TDOA fra. Dette resulterer i tre hyperbler mot hver av radarene. I figuren ses det at de to radarposisjonene blir bestemt ved at tre hyperbler skjærer hverandre. Samtidig er det tre posisjoner hvor to hyperbler skjærer hverandre. For at de riktige posisjonene skal skille seg ut, er det derfor viktig at det er flere hyperbler som skjærer i disse posisjonene enn i de gale posisjonene. Merk at det også her vil være «speil»-løsninger, altså løsninger på grunn av symmetrien i hyperblene.



Figur 5.3: To radarer bestemt ved hyperbler fra tre ulike posisjoner.

5.1.2 Stokastiske målinger

Ankomsttidene til pulsene fra navigasjonsradarene vil være støybelagt. Antar i oppgaven at ankomsttidene er støybelagt med additivt støy. Ankomsttiden til en puls i sensor j, utsendt fra en radar i i tidspunkt t_k, beregnes ved å legge til additivt støy på formel (5.1), og blir

$$t_k^j = t_k + \frac{r_k^{ij}}{c} + v_k^j, \qquad v_k^i \sim p_v(v), \tag{5.4}$$

hvor v_j er additiv støy på ankomsttiden i sensor j med fordeling $p_v(v)$. Ettersom støyen på begge ankomsttidene er additivt, vil målelikninga bli formel (5.3) med additivt støy. Målelikninga blir [6]

$$z_k = \frac{1}{c}\left(r_k^{i2} - r_k^{i1}\right) + w_k = h_k(\underline{x}) + w_k, \qquad w_k \sim p_w(w), \tag{5.5}$$

hvor $h_k(\underline{x})$ er den sanne TDOA mellom ankomsttiden i sensor 1 og 2, og w_k er summen av de to stokastiske variablene v_k^1 og v_k^2 med fordeling $p_w(w)$. Antar at støyen til ankomsttidene i sensor 1 og sensor 2 er stokastisk uavhengig. Fordelingen til støyen på TDOA er da summen av fordelingen for støyen på ankomsttidene i sensor 1 og sensor 2; og får middelverdi

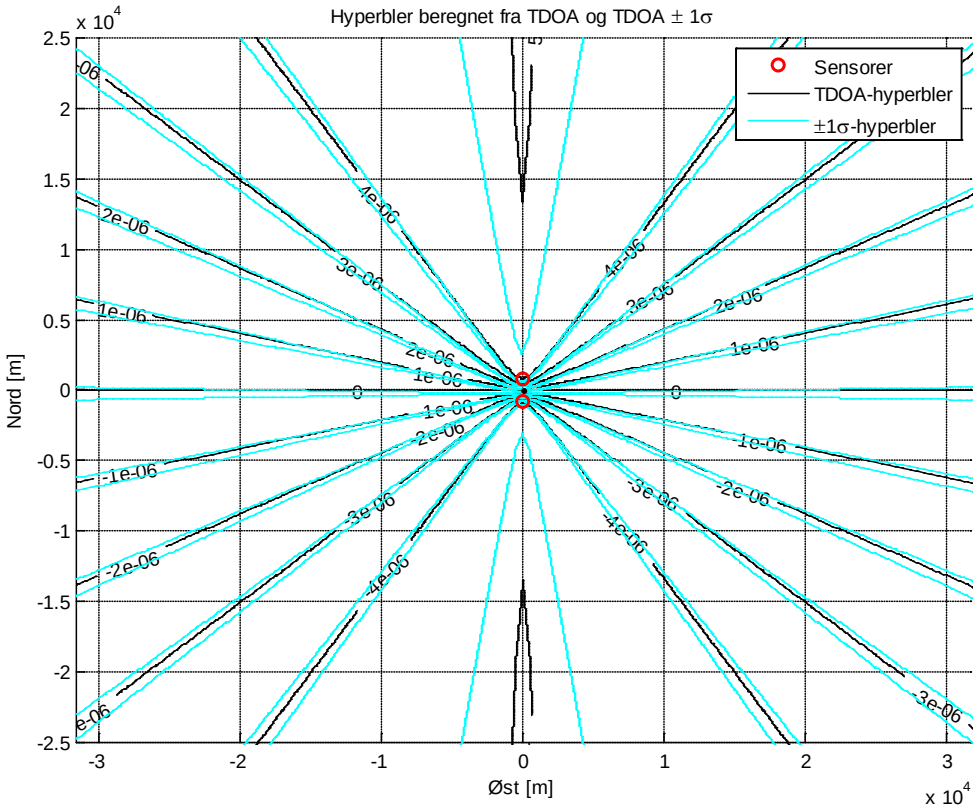
$$m_w = m_{v1} + m_{v2}, \tag{5.6}$$

hvor m_{v1} og m_{v2} er middelverdien til hhv. v_k^1 og v_k^2, og standardavvik

$$\sigma_w = \sqrt{\sigma_{v1}^2 + \sigma_{v2}^2}, \tag{5.7}$$

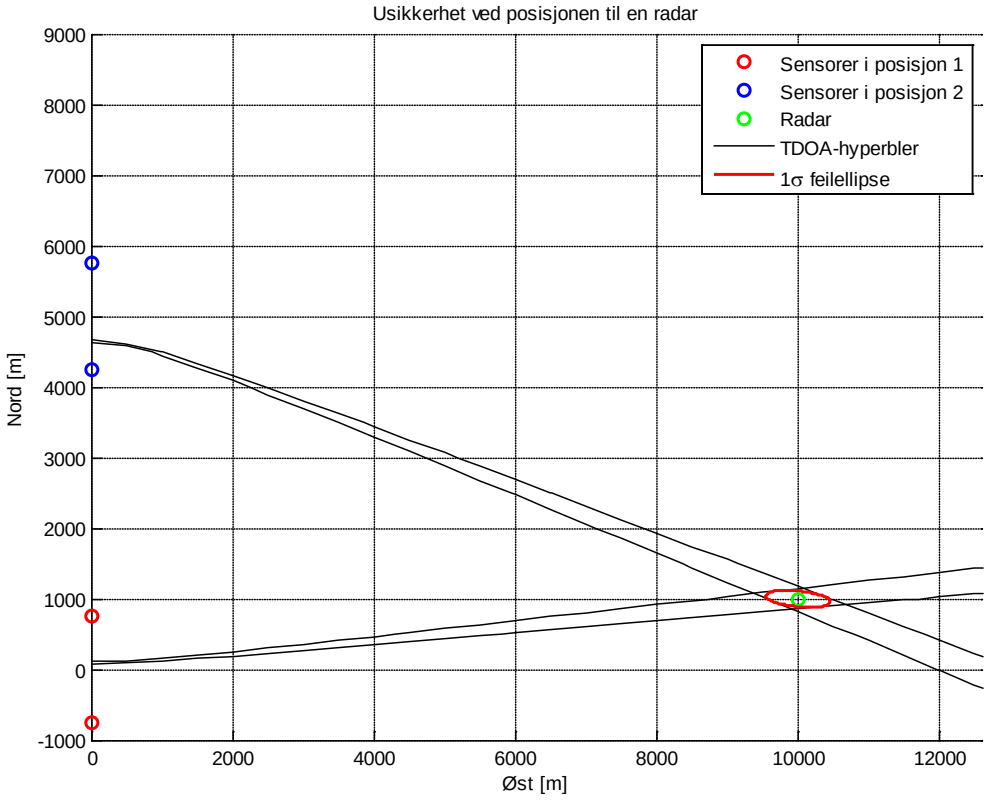
hvor σ_{v1} og σ_{v2} er standardavviket til hhv. v_k^1 og v_k^2.

Figur 5.4 viser hyperbler for noen utvalgte TDOA-er hvor det er lagt til pluss/minus et standardavvik. Som figuren viser øker spredningen til hyperblene, for en gitt TDOA, når avstanden til sensorene øker.



Figur 5.4: Hyperbler på jordoverflaten for utvalgte TDOA og TDOA pluss/minus 1σ (70,71 ns). Sensorene har en innbyrdes avstand på 1500 m og en høyde på 500 m.

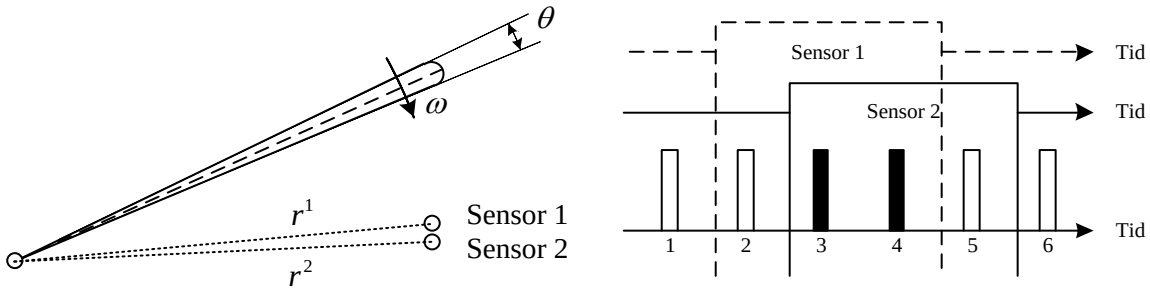
Når ankomsttidene og dermed TDOA-ene er støybelagt, vil ikke en radar nødvendigvis ligge i skjæringspunktet mellom to hyperbler. Figur 5.5 viser to sensorer som først mottar en puls i en posisjon og deretter flyttes for å motta en ny puls. Situasjonen er lik som i Figur 5.2, men her er det lagt på pluss/minus et standardavvik på de sanne TDOA-ene. Usikkerheten i TDOA fører til en usikkerhet i radarposisjon. For radarposisjonen er det tegnet opp en 1σ feilellipse som angir posisjonsusikkerheten til radaren. 1σ feilellipsen vil ligge i området begrenset av hyperblene hvor det er lagt til pluss/minus et standardavvik. Dersom vinkelen mellom hyperblene, sett fra radaren, øker, blir området begrenset av hyperblene smalere i avstand, og nøyaktigheten i avstand øker.



Figur 5.5: Posisjonen til en radar bestemt av hyperbler hvor det er lagt til pluss/minus 1σ på TDOA (70,71 ns). Sensorene har en innbyrdes avstand på 1500 m og en høyde på 500 m.

5.2 TDOA mellom flere pulser

Når UAS-ene flyr over et område med flere navigasjonsradarer vil sensorene bli bombardert av pulser. Figur 5.6 viser to figurer; a) en roterende radar (med lobebredde θ og vinkelhastighet ω) og to sensorer, og b) utsendte pulser fra en radar hvor noen av pulsene mottas i de to sensorene. Av figur a) ses det at hovedloben til radaren vil treffe sensor 1 først. Dette resulterer i at sensor 1 mottar pulser som sensor 2 ikke mottar, illustrert ved puls 2 i figur b). Ved et senere tidspunkt peker hovedloben på begge sensorene. Da mottas de samme pulsene i både sensor 1 og sensor 2, men til forskjellig tid. Dette er illustrert ved puls 3 og 4. Litt senere peker hovedloben kun på sensor 2, og sensoren mottar pulser som ikke mottas av sensor 1.



Figur 5.6: a) En roterende navigasjonsradar og to sensorer [5]. b) Pulstog fra en radar hvorav noen av pulsene mottas av sensor 1 og sensor 2 [5].

Når sensorene mottar pulser fra flere radarer, kan i utgangspunktet alle pulser i den ene sensoren høre sammen med alle pulser i den andre radaren. Noen pulser mottas av begge sensorene, mens andre pulser mottas bare i en av sensorene. Antar at sensor 1 mottar pulser i tidspunktene $T^1 = \{t_1^1, t_2^1, \dots, t_n^1, \dots, t_N^1\}$ og sensor 2 i tidspunktene $T^2 = \{t_1^2, t_2^2, \dots, t_m^2 \dots, t_M^2\}$. For hver ankomsttid i de to sensorene er også sensorens posisjon kjent. TDOA mellom alle pulser i sensor 1 og alle pulser i sensor 2 kan beregnes ved

$$Z = [z_{mn}] = \begin{bmatrix} t_0^2 & t_0^2 & \cdots & t_0^2 \\ t_1^2 & t_1^2 & \cdots & t_1^2 \\ \vdots & \vdots & \ddots & \vdots \\ t_M^2 & t_M^2 & \cdots & t_M^2 \end{bmatrix} - \begin{bmatrix} t_0^1 & t_1^1 & \cdots & t_N^1 \\ t_0^1 & t_1^1 & \cdots & t_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ t_0^1 & t_1^1 & \cdots & t_N^1 \end{bmatrix}. \tag{5.8}$$

I formel (5.8) vil kun et mindretall av TDOA bestå av to ankomsttider til den samme pulsen, og flesteparten av TDOA-ene vil være satt sammen av ankomsttider til forskjellige pulser. Fra avsnitt 5.1.1 fås at en TDOA må ligge i intervallet $[-d/c\,, d/c]$. Elementene i formel (5.8) må derfor tilfredsstille

$$z_{mn} \leq \frac{d}{c}, \tag{5.9}$$

hvor d er avstanden mellom sensorene. Alle TDOA fra formel (5.8) som tilfredsstiller formel (5.9) plasseres i en vektor. Får da Fishermodellen (se appendiks D)

$$\underline{z} = \underline{h}(\underline{x}) + \underline{w}, \qquad \underline{w} \sim p_w(\underline{w}), \tag{5.10}$$

hvor $\underline{h}(\underline{x})$ er en ulineær vektorfunksjon av den ukjente, men konstante radarposisjonen $\underline{x}$, og $\underline{w}$ er additiv støy med fordeling $p_w(\underline{w})$. Linje k i den ulineære vektorfunksjonen gir den deterministiske TDOA mellom ankomsttiden til puls m i sensor 2 og puls n i sensor 1, og bestemmes ved formel (5.3)

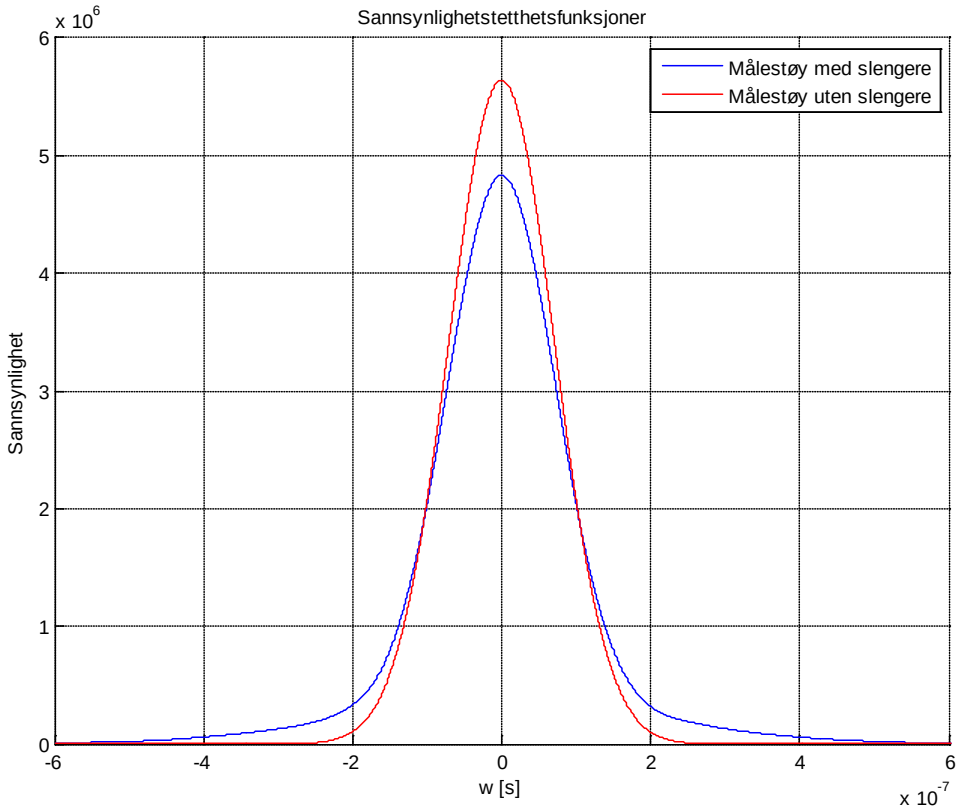
$$h_k(\underline{x}) = \frac{1}{c}\left(\left\|\underline{x} - \underline{p}_{S2}^N(m)\right\| - \left\|\underline{x} - \underline{p}_{S1}^N(n)\right\|\right). \tag{5.11}$$

hvor $\underline{p}_{S2}^N(m)$ er posisjonen til sensor 2 i ankomsttidspunkt t_m^2 og $\underline{p}_{S1}^N(n)$ er posisjonen til sensor 1 i ankomsttidspunkt t_n^1. Forhåpentligvis inneholder vektoren i formel (5.10) flest mulig TDOA som er beregnet mellom den samme pulsen i de to sensorene.

5.3 Støy og feilkilder

Ankomsttidene til radarpulsene i begge sensorene vil være støybelagt. Støyen på ankomsttidene vil i hovedsak skylles feil i ankomsttiden til pulsene, tilfeldige feil i klokker og usikkerhet i sensorens posisjon. I [2] blir det undersøkt hva feil i to flys posisjon utgjør for målenøyaktigheten i TDOA. Der vises det at dersom posisjonsusikkerheten til flyene i retning nord og øst er stokastisk uavhengig og normalfordelt med i middelverdi 0 m og standardavvik på 10 m, resulterer dette i en usikkerhet i TDOA på 47,17 ns. Det betyr at usikkerheten i posisjonen kun gir en usikkerhet i TDOA. Det antas derfor i denne oppgaven at sensorenes posisjonsusikkerhet ligger innbakt i ankomsttidenes usikkerhet.

I oppgaven er ankomsttidene først belagt med additiv gaussisk støy med middelverdi 0 ns og standardavvik 50 ns. I denne støyen ligger bl.a. feil i ankomsttiden til pulsene, tilfeldige feil i klokker, og usikkerhet i sensorens posisjon. Antar at støyen i de to sensorene er stokastisk uavhengig. Målestøyen (støyen i TDOA) blir da gaussisk med middelverdi 0 ns og standardavvik 70,71 ns (fra formel (5.6) og (5.7)). Deretter introduseres det slengere i oppgaven. Slengere er et fenomen som ofte oppstår i reelle målinger og er målinger som kan se helt gale ut. I oppgaven antas det at 10 % av ankomsttidene i hver av sensorene er slengere, mes de resterende 90 % er «vanlige» målinger. Dette fører til at omtrent 20 % av TDOA-ene er slengere. Slengerne fordeles tilfeldig utover alle ankomsttidene i de to sensorene, og trekkes fra en gaussisk fordeling med middelverdi 0 ns og standardavvik 200 ns. Også her antas det at ankomsttidene i de to sensorene er stokastisk uavhengig. Figur 5.7 viser fordelingen for målestøy uten slengere (rød) og målestøy hvor omtrent 20 % er slengere (blå). I figuren ses det at fordelingen for målestøy med slengere er noe bredere enn fordelingen målestøy uten slengere, og sannsynligheten for å trekke støy som ligger langt fra middelverdien er derfor noe høyere.



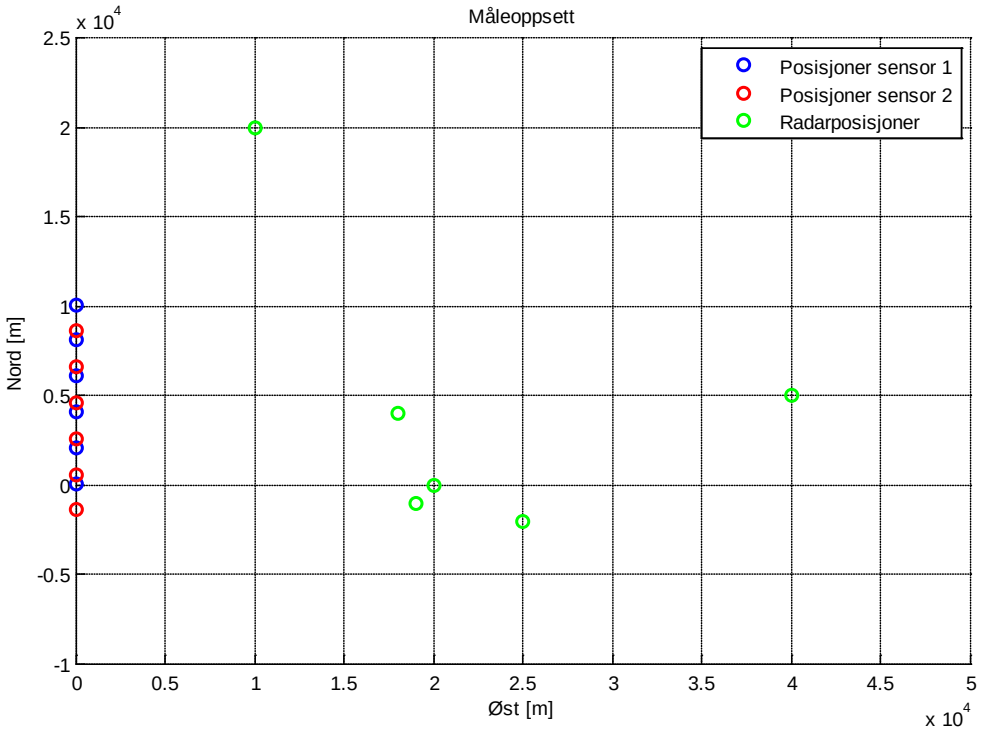
Figur 5.7: Ikke-gaussisk fordeling for målestøy med slengere og gaussisk fordeling for målestøy uten slengere.

Fordelingen for målestøy med slengere er funnet ved å trekke tilstrekkelig antall ankomsttider, hvorav 10 % av ankomsttidene er trukket fra en gaussisk fordeling med middelverdi 0 ns og standardavvik 200 ns og de resterende 90 % av ankomsttidene er trukket fra en gaussisk fordeling med middelverdi 0 ns og standardavvik 50 ns. Deretter er TDOA beregnet, og sannsynlighetstetthetsfordelingen funnet ved hjelp av et histogram.

6 Pulsassosiasjon og geolokaliseringsalgoritme

Pulsassosiasjon og geolokaliseringsalgoritmen assosierer pulser fra samtidige radarer og geolokaliserer disse. Algoritmen er delt inn i to deler. I avsnitt 6.1 beskriver hvordan TDOA mellom pulser fra flere radarer beregnes, og hvordan TDOA fra samme radar assosieres. I avsnitt 6.2 geolokaliseres radarene vha. TDOA-ene fra avsnitt 6.1. Geolokaliseringen går først ut på å finne mulige radarposisjoner, og deretter begynner en estimator å estimere radarposisjonene. Algoritmen i kapittelet bygger på løsninger beskrevet i [2]. MATLAB-koden til pulsassosiasjon og geolokaliseringsalgoritmen ligger i appendiks I.1.

Algoritmen er laget mht. scenarioet med to UAS-er, ettersom UAS-ene trolig mottar pulser fra færre radarer enn to satellitter. Signalmiljøet er da enklere enn for satellitt-scenarioet, og det er da trolig enklere å lage en algoritme for pulsassosiasjon og geolokalisering. Deretter er algoritmen tilpasset scenarioet med to satellitter. Underveis i kapittelet benyttes det simulerte ankomsttider fra pulssimulatoren beskrevet i kapittel 4. I simulatoren flyr to UAS-er nordover med en hastighet på 20 m/s, i en høyde på 500 m og med en innbyrdes avstand på 1500 m. Seks roterende navigasjonsradarer er plassert utover jordoverflaten. De seks radarene er identiske mht. strålingsdiagram og sendereffekt, men har ulik PRI, rotasjonstid og initiell pekeretning. Signalmiljøet i de to sensorene blir da forholdsvis realistisk, hvor sensorene noen ganger pent og pyntelig mottar pulser fra en radar av gangen, mens andre ganger mottar pulser fra flere radarer samtidig. Ettersom UAS-ene flyr forholdsvis langsomt, er det valgt å utføre flere måleintervall ulike posisjoner. I simuleringen gjennomføres det seks måleintervall i tidspunktene 0-10 s, 100-110 s, 200-210 s, 300-310 s, 400-410 s, og 500-510 s. Figur 6.1 viser oppsettet av UAS-ene og de seks navigasjonsradarene som er benyttes i kapittel 4. I figuren er sensorenes posisjoner tegnet opp i hvert av de seks tidsintervallene.



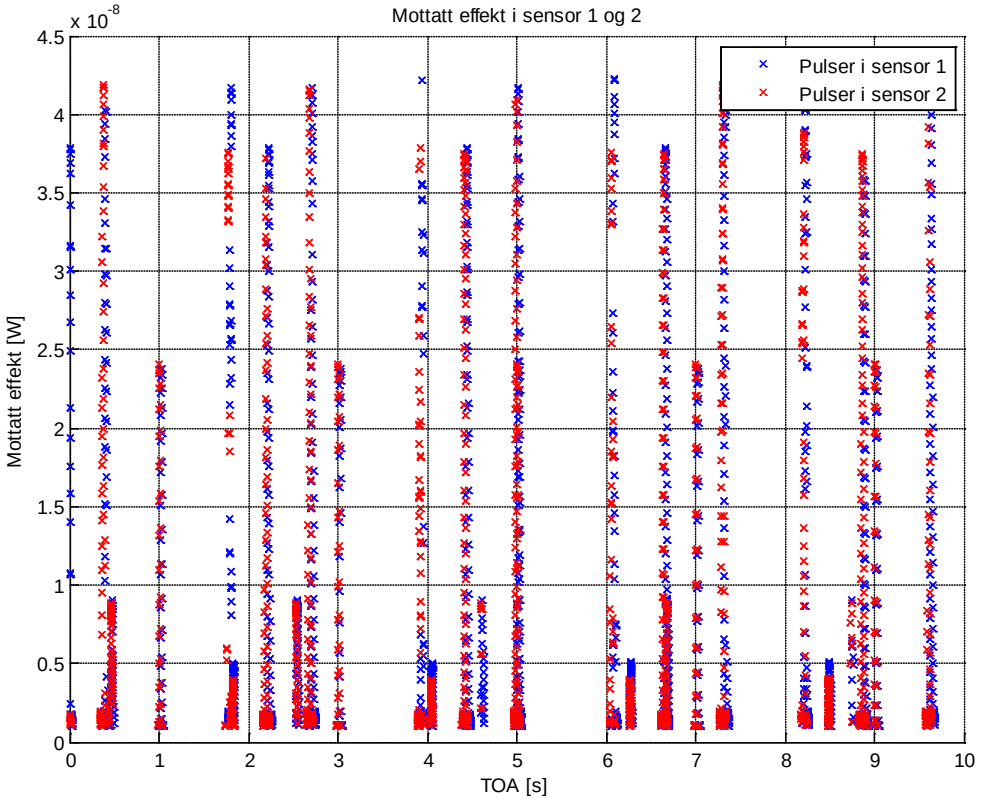
Figur 6.1: UAS-posisjoner ved seks måleintervall og seks navigasjonsradarer.

6.1 Pulsassosiasjon ved TDOA

Begge sensorene vil motta en rekke pulser fra flere radarer. Mellom ankomsttidene til pulsene beregnes det tidsdifferansen (TDOA). Avsnitt 6.1.1 beskriver hvordan algoritmen beregner TDOA mellom ankomsttiden til pulser, og hvordan TDOA fra galt sammensatte pulser lukes bort. I avsnitt 6.1.2 beskrives hvordan algoritmen assosierer pulser som trolig kommer fra samme radar ved å skille pulsene vha. TDOA.

6.1.1 Beregning av TDOA

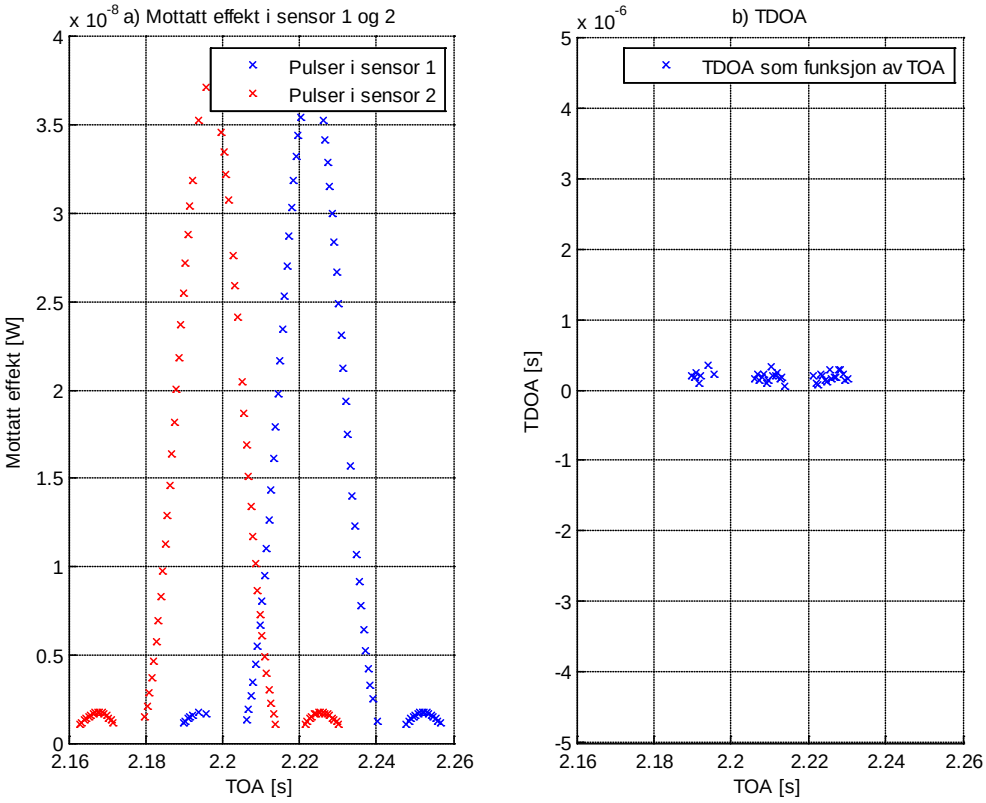
Sensorene vil motta pulser fra flere radarer, og for hver puls beregnes ankomsttidspunktet. Figur 6.2 viser mottatte pulser fra de seks roterende radarene i løpet av ti sekunder. Langs y-aksen vises den mottatte effekten til pulsene, og langs x-aksen vises ankomsttiden til pulsene (TOA). Blå pulser er mottatt i sensor 1 og røde pulser i sensor 2. I simuleringen mottas det hovedsakelig pulser fra radarenes hovedlober og et par sidelober. For flere av hovedlobene ses det at de mottatte pulsene i sensor 1 og sensor 2 er forskjøvet litt i tid, altså at pulsene mottas i den ene sensoren før den andre sensoren.



Figur 6.2: Mottatte pulser fra seks navigasjonsradarer i to sensorer ved 10 s simulering.

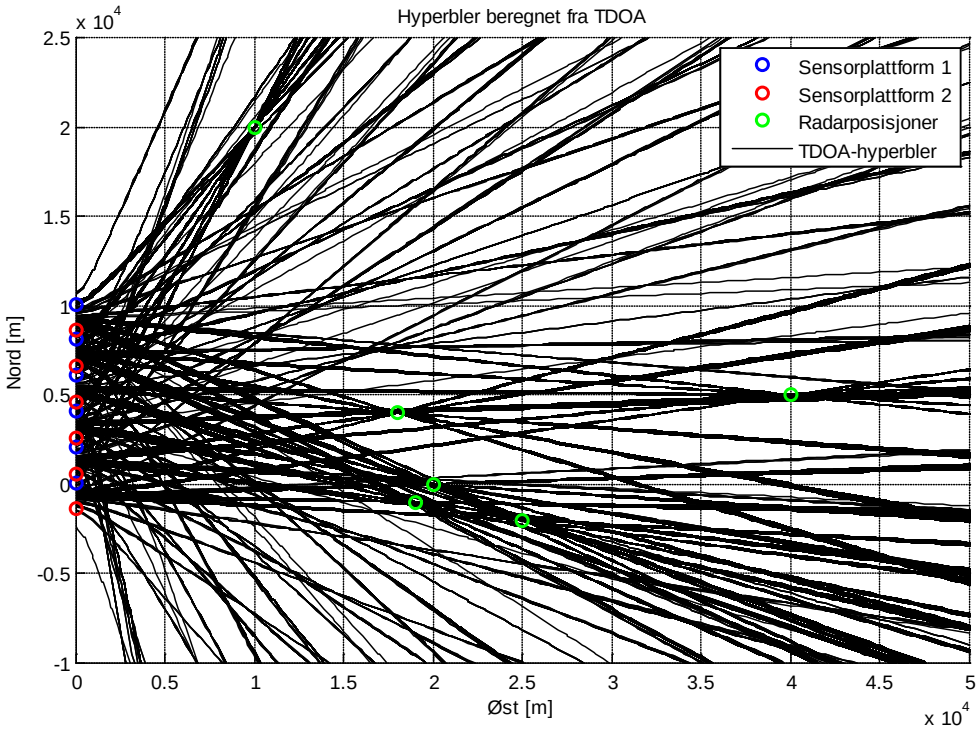
I oppgaven benyttes ingen annen informasjon enn TDOA til å lokalisere radarene. Problemet blir å finne hvilke pulser som kommer fra samme radar. Starter derfor å beregne TDOA mellom alle ankomsttider i sensor 1 og sensor 2 ved formel (5.8). Store deler at TDOA-ene vil lukes bort ved formel (5.9), som sier at en TDOA må ligge i intervallet $[-d/c, d/c]$, hvor $d/c \approx 5$ µs (gitt 1500 m mellom sensorene). Resten av TDOA-ene plasseres i en vektor, og målelikninga fra formel (5.10) fås. Ettersom de fleste av TDOA-ene ikke tilfredsstiller formel (5.9), vil vektoren stort sett inneholder TDOA mellom pulser fra samme hovedlobe, eller TDOA mellom pulser fra to radarer som begge peker mot sensorene.

Figur 6.3 viser to figurerer; a) viser et utsnitt av en hovedlobe fra Figur 6.2, og b) TDOA beregnet mellom ankomsttidene i de to sensorene fra den samme hovedloben. x-aksen i begge figurene er lik, mens y-aksen i figur b) viser TDOA i intervallet [-5 µs, 5 µs]. I tilfellet fra figuren beregnes det TDOA mellom pulsene fra radaren når hovedloben peker på sensor 2 og sideloben peker på sensor 1, når hovedloben peker på begge sensorene, og når hovedloben peker på sensor 1 og sideloben på sensor 2. I dette tilfeller beregnes det 36 TDOA mot den samme radaren, hvorav 14 av TDOA-ene beregnes mellom pulser når hovedloben peker på begge sensorene.

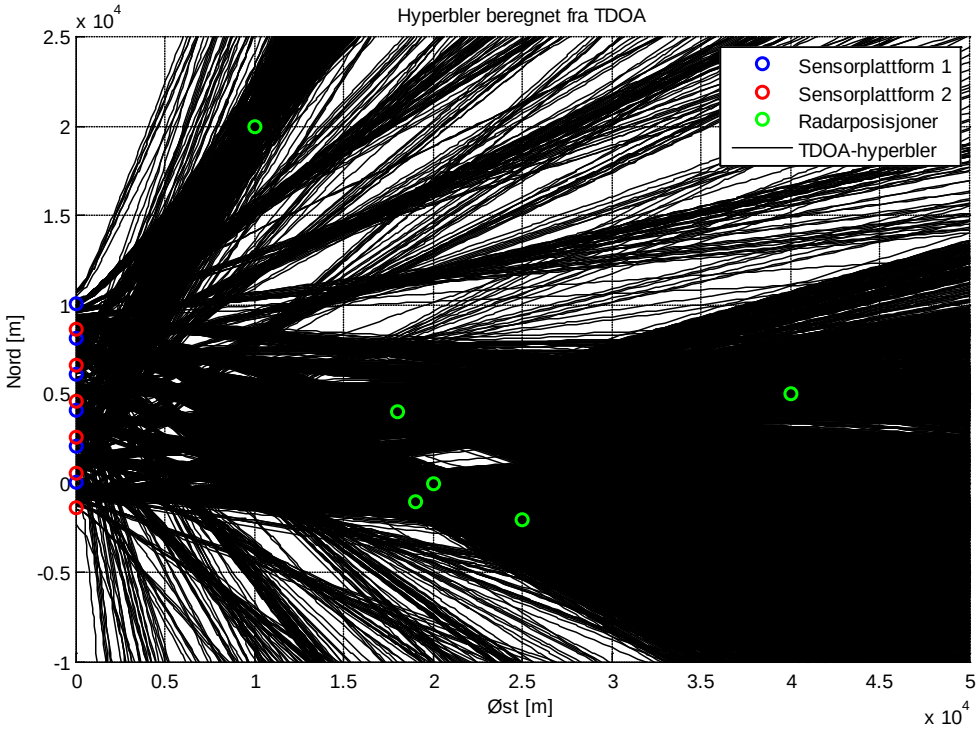


Figur 6.3: a) Mottatte pulser fra en hovedlobe til en radar i sensorene. b) TDOA mellom pulser i sensor 1 og 2.

Av TDOA-ene kan hyperbler beregnes. Figur 6.4 viser hyperbler beregnet fra TDOA hvor det ikke er lagt til støy. Figur 6.5 viser hyperbler til de samme TDOA-ene, men hvor det er lagt til gaussisk støy med middelverdi 0 ns og standardavvik 70,71 ns. Fra avsnitt 5.1 kunne radarenes posisjoner bestemmes ved at mange hyperbler skjærer i det samme punktet. I Figur 6.4 ses det tydelig at mange hyperbler skjærer hverandre i radarposisjonene. Samtidig inneholder figuren også mange hyperbler som ikke går gjennom noen radarposisjoner. Disse hyperblene skylles TDOA-er som er beregnet mellom pulser fra to ulike radarer. I Figur 6.5 er det vanskeligere å bestemme radarposisjonene. Allikevel er det mange hyperbler som går i retning av radarposisjonene.



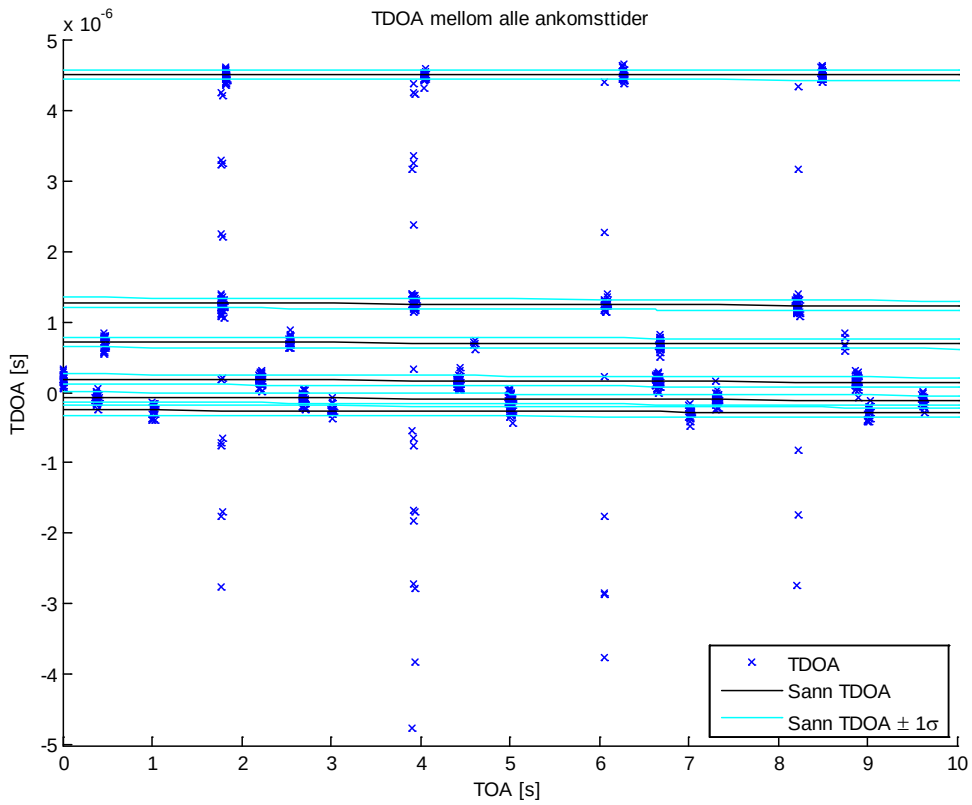
Figur 6.4: Hyperbler beregnet av TDOA uten støy.



Figur 6.5: Hyperbler beregnet av TDOA med støy.

6.1.2 Midling av TDOA fra samme radar

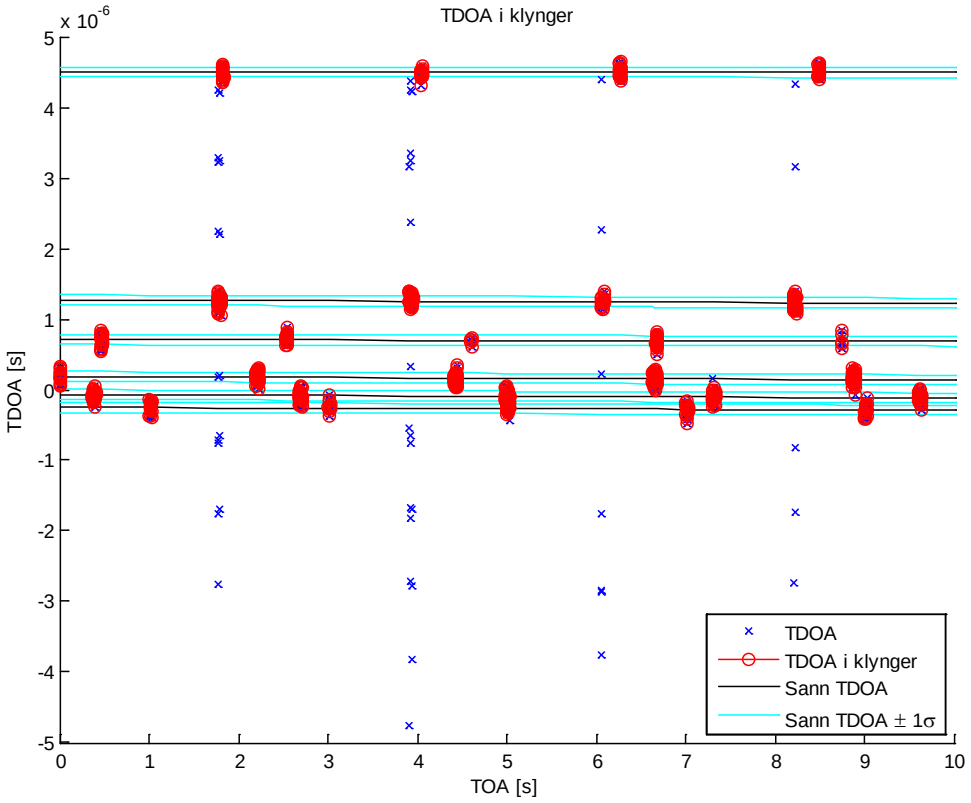
Spredningen i TDOA fører til stor spredning av hyperblene, noe som ses tydelig i Figur 6.5, og det kan derfor være vanskelig å bestemme posisjonene til radarene som sendte ut pulsene. Figur 6.6 viser TDOA mellom pulser som funksjon av ankomsttiden til pulsene fra Figur 6.2. I figuren viser y-aksen TDOA og x-aksen ankomsttiden til pulsene som inngår i TDOA-ene. I figuren er det også tegnet opp de sanne TDOA-ene (TDOA mot radarposisjoner hvor det ikke er lagt til støy) til de seks radarene og de sanne TDOA-ene pluss/minus et standardavvik. I figuren er det mange tette klynger med TDOA-er som inneholder TDOA-er fra samme radar. Klyngene gjentas typisk med 2-3 s intervall, noe som stemmer med at radarene roterer med en rotasjonstid på 2-3 s. I figuren er det også en del TDOA som ikke ligger i klynger, eller som ikke ligger i nærheten av noen sanne TDOA. Dette er TDOA-er som er satt sammen av pulser fra to ulike radarer, eller TDOA-er som er satt sammen av ulike pulser fra samme radar.



Figur 6.6: TDOA mellom pulser i sensor 1 og sensor 2 og sann TDOA til radarene.

Ved å finne klynger med TDOA som tilhører samme radar og midle disse, vil spredningen til hyperblene reduseres. Ønsker da å finne klynger med TDOA-er som kommer fra samme hovedlobepassering til hver av radarene. Ettersom TDOA-ene i hver klynge skal midles, må også sensorenes posisjoner midles. For at sensorenes posisjon, og dermed endringen av TDOA mot samme radar, ikke skal endres for mye i løpet av tiden TDOA-ene midles, deles TDOA-ene opp i intervaller etter når pulsene er mottatt. Velger å dele opp ankomsttidene, og dermed TDOA-ene, opp i intervaller på 100 ms. Sensorene vil da maksimalt flytte seg 2 m. Forhåpentligvis havner alle TDOA-ene fra en hovedlobepassering til en radar innenfor intervallet på 100 ms. Ettersom hvert intervall kan inneholde TDOA fra flere ulike radarer eller galt sammensatte TDOA-er,

benyttes det klyngeanalyse til å fordele TDOA-ene i klynger. Det er valgt å benytte hierarkisk klyngeanalyse (for med informasjon om hierarkisk klyngeanalyse, se appendiks E), ettersom antall klynger ikke er kjent. Den hierarkiske klyngeanalysen fungerer ved at den starter med like mange klynger som TDOA. Deretter slår den sammen de to nærmeste klyngene, helt til det bare gjenstår en klynge eller et kriterium er oppfylt. I oppgaven stopper klyngeanalysen når avstanden mellom alle klyngene er større enn 212,13 ns (3σ). Klyngeanalysen gir da klynger med en eller flere TDOA. I algoritmen er det satt at hver klynge minimum må inneholder fire TDOA. Hvis klyngen inneholder mindre enn fire TDOA, fjernes klyngen og TDOA-ene som inngår antas å være galt sammensatte TDOA. Figur 6.7 viser samme figur som Figur 6.6, men hvor det er benyttet klyngeanalyse. Alle TDOA-ene som tilhører klyngene er tegnet med en rød sirkel. Som figuren viser, har klyngeanalysen funnet en rekke klynger med TDOA-er. Samtidig er også en del av TDOA-ene utelatt fra klyngene (blå kryss uten rød ring rundt), ettersom eventuelle klynger inneholder færre enn fire TDOA. Disse utelatte TDOA-ene vil enten være TDOA som er satt sammen av riktige pulser og som burde vært en del av klyngene, eller TDOA-er som er galt sammensatt.



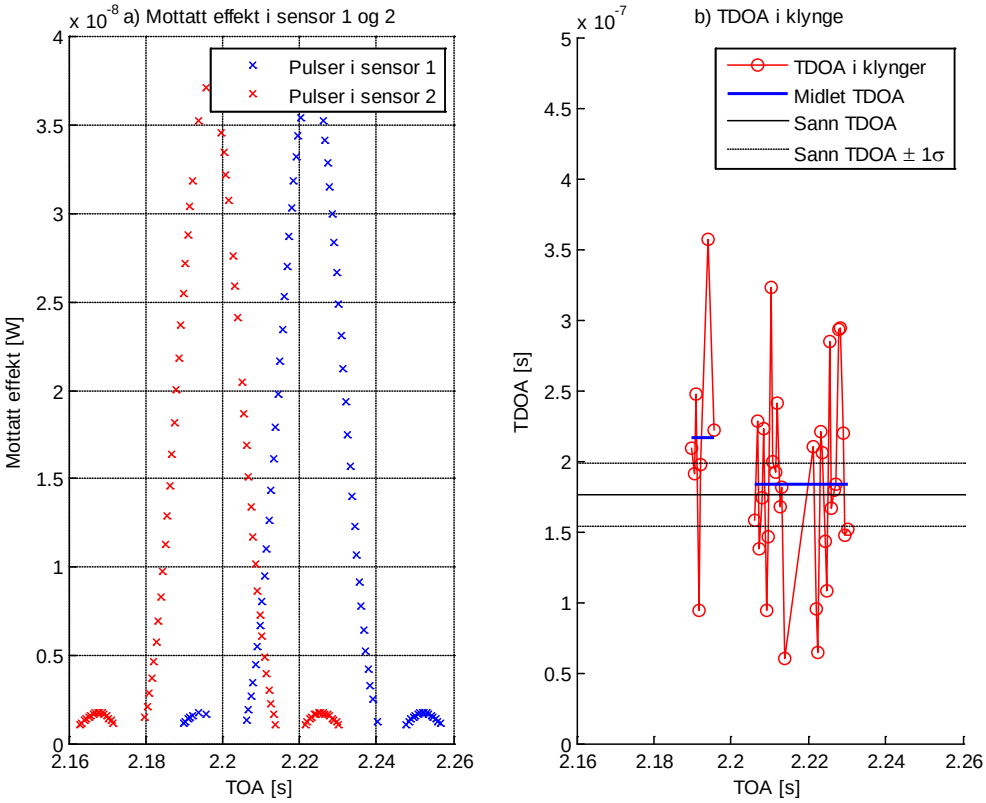
Figur 6.7: TDOA i klynger som funksjon av ankomsttiden.

Ved å benytte klyngeanalyse på TDOA-ene, assosieres pulsene fra de ulike radarene. Ideelt sett tilsvarer hver klynge da pulser fra en hovedlobepassering til en radar. Klyngene med TDOA-er midles, sammen med tilhørende sensorposisjoner, og benyttes videre i oppgaven. Alle TDOA-er som ikke tilhører noen klynge, antas å være galt sammensatte TDOA-er og fjernes.

Ved å midle TDOA-ene i hver klynge, reduseres standardavviket til disse midlede TDOA-ene faktor på $\sqrt{N}$ i forhold til de ikke-midlede TDOA-ene, hvor N er antall TDOA som midles. Dette forutsetter riktignok at støyen på ankomsttidene i hver sensor er stokastisk uavhengig. Antall

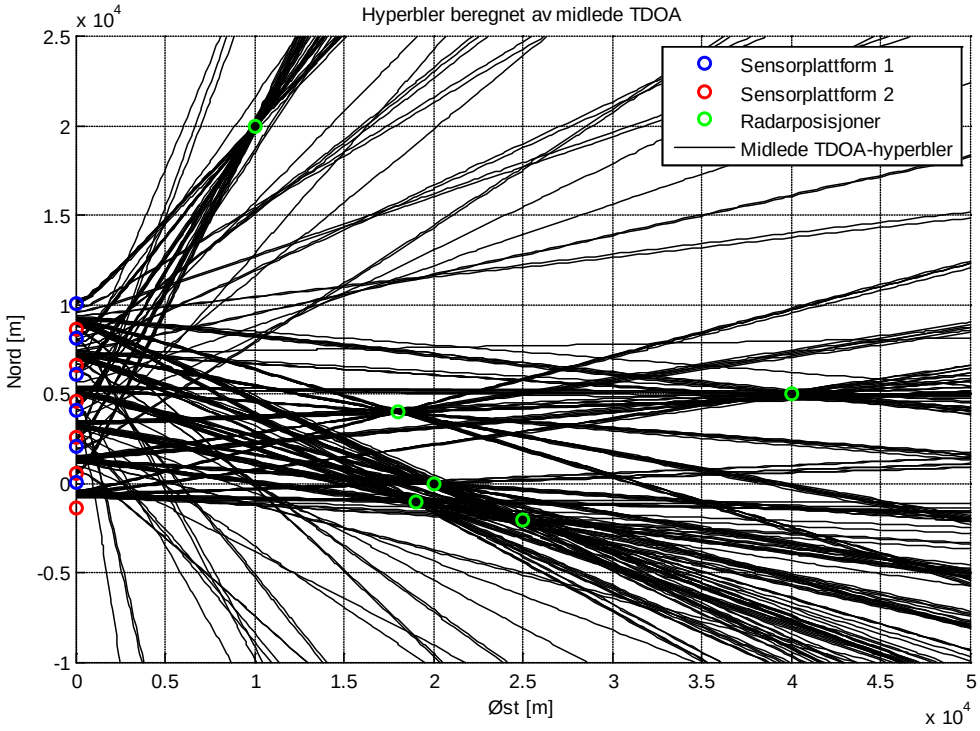
TDOA som midles vil i midlertidig variere fra klynge til klynge, men typisk består hver klynge av 10 TDOA og oppover. For å ha en verdi på standardavviket til de midlede TDOA-ene settes antall TDOA per klynge til N = 10.

Figur 6.8 viser to figurerer; a) viser et utsnitt av en hovedlobe fra Figur 6.2, og b) TDOA beregnet mellom ankomsttider fra den samme hovedloben. Figuren viser det samme tilfellet som i Figur 6.3, men her er det kjørt klyngeanalyse på TDOA-ene. I figur b) er TDOA-ene som er inneholdt i en klynge tegnet med en rød sirkel. Mellom TDOA-ene i klyngen er det trukket en rød linje for å vise hvilke TDOA-er som midles. Hovedlobepasseringen i figuren deles tilfeldigvis opp i to 100 ms intervall, noe som resulterer i at en hovedlobepassering fra en radar gir to midlede TDOA (blå linje). I figuren er det også tegnet den sann TDOA mot radaren som sendte ut pulsene og sann TDOA pluss/minus et standardavvik for de midlede TDOA (σ_m = 22,63 ns). I figuren ses det at middelverdien til TDOA-ene i klyngen mot venstre ligger innfor pluss/minus et standardavvik, mens middelverdien til TDOA-ene i klyngen mot høyre ligger utenfor.



Figur 6.8: a) Mottatte pulser fra en hovedlobe til en radar i sensorene. b) TDOA-er i to klynger og midlede TDOA-er.

Figur 6.9 viser hyperbler beregnet av de midlede TDOA-ene. Ved å sammenlikne figuren med Figur 6.5 ses det at både antall hyperbler og spredningen til hyperblene er redusert drastisk. I figuren tilsvarer hver hyperbel midlede TDOA fra en hovedlobepassering til en radar. Hyperblene som ikke går mot noen radar skylles i hovedsak at klyngeanalysen har opprettet klynger for TDOA-er som er gale. Ved å øke minimum antall TDOA-er per klynge, vil mange av de gale hyperblene forsvinne. Samtidig kan også noen midlede TDOA-er som består av få, men riktig sammensatte TDOA-er, også forsvinne. I figuren skiller radarposisjonene seg ut ved at svært mange hyperbler skjærer hverandre innenfor et lite område, og at alle seks måleintervallene gir hyperbler mot radarposisjonene.



Figur 6.9: Hyperbler beregnet fra de midlede TDOA.

6.2 Geolokalisering av radarer

I avsnitt 6.1 er det beskrevet hvordan algoritmen beregner TDOA mellom pulsene, og hvordan TDOA-ene danner klynger med andre TDOA-er fra samme radar. Målet var da at klyngene inneholdt alle TDOA-ene fra en hovedlobepassering til en radar. TDOA-ene i klyngene ble deretter midlet, slik at spredningen ble redusert. I resten av rapporten benyttes det bare disse midlede TDOA-ene (heretter bare referert til som TDOA) til å estimere posisjonene til radarene som sendte ut pulsene.

Når radarposisjonene skal estimeres, gjøres dette på bakgrunn av TDOA-ene. Det er da viktig at estimatoren kun benytter TDOA mellom pulser sendt ut av nettopp den radaren som skal lokaliseres, altså at TDOA-ene fra samme radar assosieres, hvis ikke vil estimatoren gi et galt estimat. [2] løste dette ved å legge ut et rektangulært grid med initielle posisjoner. I hver initielle posisjon ble det valgt ut TDOA som kunne komme fra denne posisjonen. Deretter tok en estimator utgangspunkt i de initielle posisjonene og estimerte posisjoner med de tilhørende TDOA-ene, et for hvert gridpunkt. Deretter ble det analysert hvilke av estimatene som var riktige og hvilke som var gale.

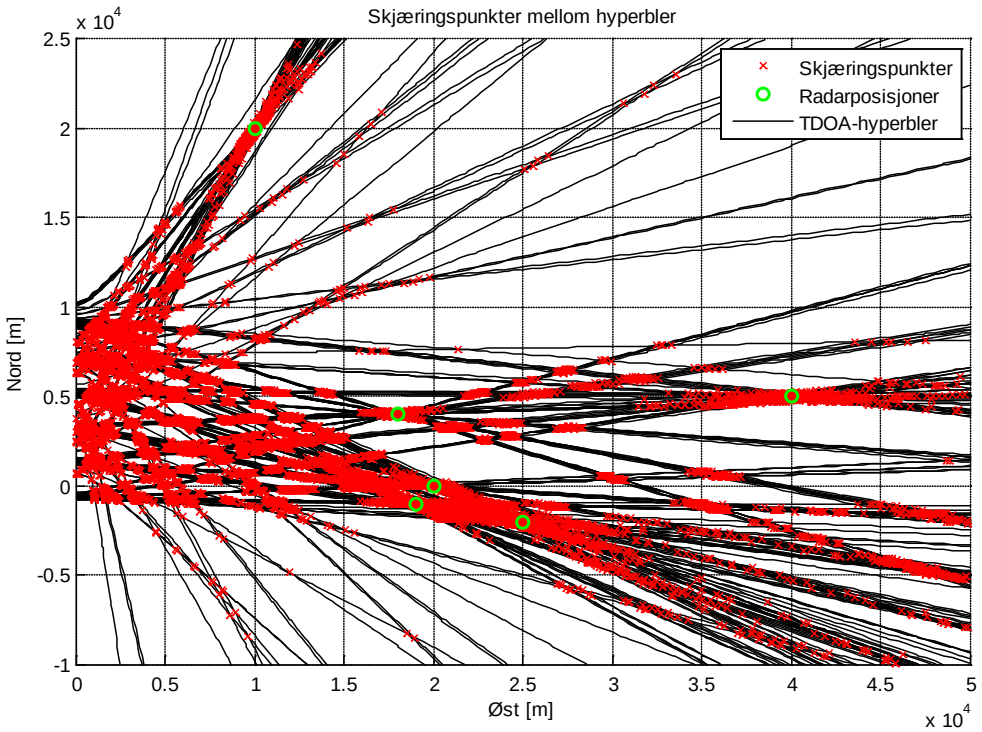
Algoritmen i denne oppgaven bygger på metoden fra [2]. Først legges det ut en rekke initielle posisjoner. For hver initielle posisjon velges det ut de TDOA-er som kan komme fra den posisjonen dersom en radar var plassert der. For å begrense antall estimat og for å begrense

regnetiden, er algoritmen basert på en såkalt «greedy»-filosofi. En greedy-filosofi søker beste lokale løsning i håp om at dette også skal gi den beste globale løsningen. Når en posisjon er estimert, antas de derfor at TDOA-ene som ble benyttet i estimatoren tilhører denne posisjonen. Disse brukte TDOA-ene fjernes slik at andre estimat ikke kan benytte disse. En del av algoritmen går dermed ut på å finne hvilke av de initielle posisjonene som skal estimeres først. Antall estimat vil ved denne metoden begrenses ved at de initielle posisjonene etter hvert ikke har noen tilhørende TDOA-er.

Avsnittet er delt inn i tre deler; i avsnitt 6.2.1 beskriver hvordan de initielle radarposisjonene bestemmes, i avsnitt 6.2.2 beskrives hvordan de ulike initielle posisjonene rangeres og hvordan de estimerte posisjonene beregnes, mens i avsnitt 6.2.3 ses det på hva som skiller riktige estimat fra gale estimat.

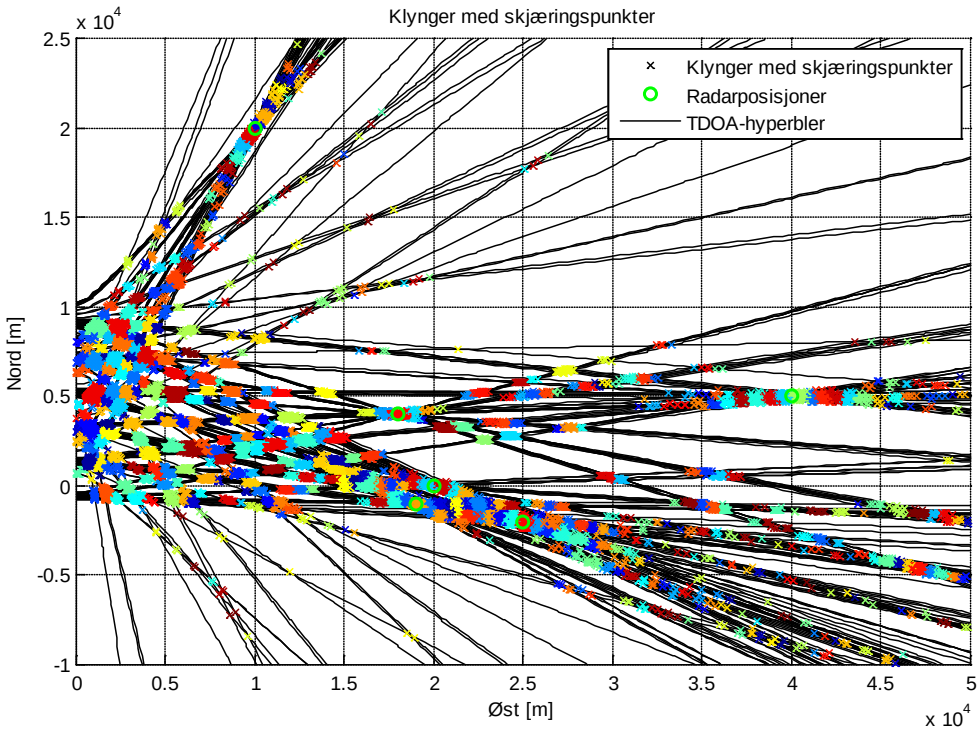
6.2.1 Initielle posisjoner

De initielle posisjonene benyttes som startposisjoner for estimering av radarposisjonene. [2] la ut et rektangulært grid med initielle posisjoner. Ulempen med et rektangulært grid er at det bør være tett hvor radarene ligger tett, mens det ikke trenger å være initielle posisjoner hvor det ikke ligger noen radar. I denne oppgaven benyttes en annen metode for å opprette initielle posisjoner. Fra avsnitt 5.1 fås det at en radar ligger i skjæringspunktet mellom to hyperbler når det ikke er lagt til støy. I det stokastiske tilfellet vil en radar ligge i et område hvor mange hyperbler skjærer hverandre. De initielle posisjoner plasseres ut der tettheten av skjæringspunkter er høyt. Figur 6.10 viser skjæringspunkter (røde kryss) mellom hyperbler fra to ulike måleintervall.



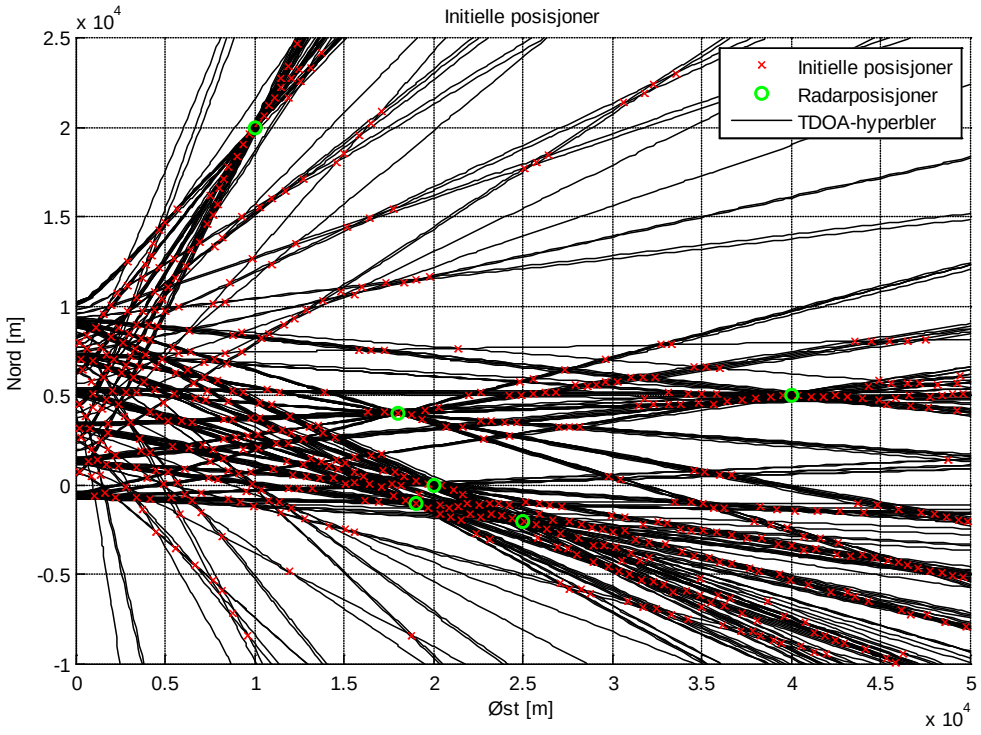
Figur 6.10: Skjæringspunkter mellom to hyperbler fra to ulike måleintervall.

Det vil trolig ligge en radar i områder med mange skjæringspunkter. For å samle disse områdene, benyttes det klyngeanalyse. Tyngdepunktet, eller middelverdien, til hver klynge vil da fungere som initielle posisjoner. For å bestemme klynger med skjæringspunkter, benyttes hierarkisk klyngeanalyse. Klyngeanalysen starter med like mange klynger som antall skjæringspunkter. Deretter beregnes den euklidske avstanden mellom middelverdien til alle klyngene, og de to nærmeste klyngene slås sammen. Slik fortsetter klyngeanalysen inntil alle skjæringspunktene ligger i samme klynge, eller at et kriterium er oppfylt. Utfordringen med klyngeanalysen er å få klynger med tyngdepunktene så nært radarene som mulig. For at et tyngdepunkt skal ligge i nærheten av en radar, må klyngen inneholde kun skjæringspunkter mellom hyperbler som tilhører nettopp den radaren. Klyngene bør derfor være små nok til å skille skjæringspunkter fra to radarer som ligger tett, og samtidig store nok til at tyngdepunktene blir riktig (ligger så nært radarene som mulig). Prøving og feiling har resultert i at en to klynger kun slås sammen dersom den euklidske avstanden mellom senter av klyngene er mindre enn 500 m. Figur 6.11 viser klynger med skjæringspunktene fra Figur 6.10. På grunn av mangel på farger er de samme fargene benyttet flere ganger, men da bør det komme frem av avstanden mellom klyngene at de ikke tilhører samme klynge.



Figur 6.11: Klynger med skjæringspunkter. Kryssene med lik farge er skjæringspunkter i samme klynge.

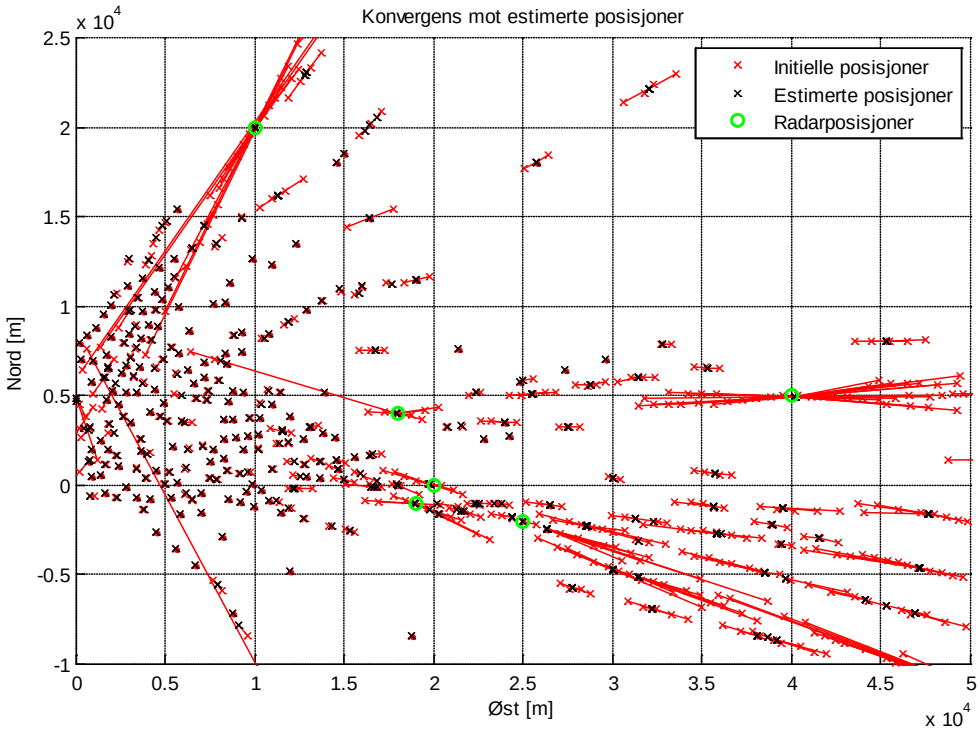
Tyngdepunktet til hver klynge er i seg selv et estimat at radarposisjonene hvor det legges vekt på områder med mange skjæringspunkter, men i oppgaven benyttes tyngdepunktene som initielle posisjoner i estimeringsalgoritmen. Figur 6.12 viser initielle posisjoner som estimatoren benytter til å beregne et estimat av radarposisjonene.



Figur 6.12: Initielle posisjoner (tyngdepunktene av klyngene).

6.2.2 Estimering av posisjoner

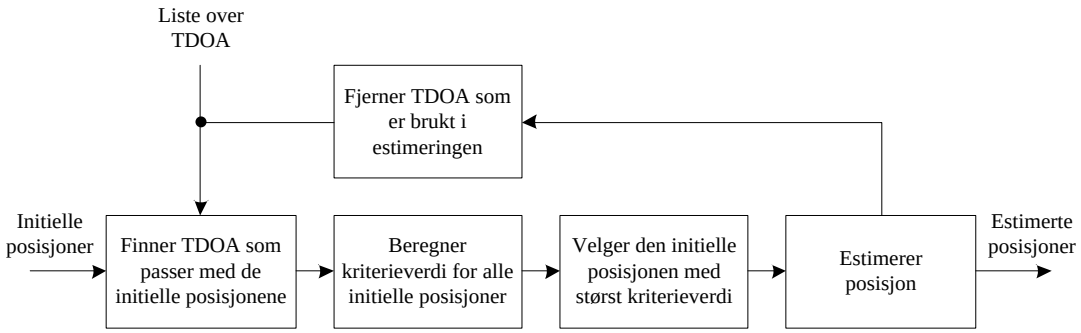
Fra avsnitt 6.2.1 fås et sett med initielle posisjoner. Disse initielle posisjonene benyttes i en søkealgoritme som søker etter et estimat av radarposisjonene. Ettersom de initielle posisjonene ligger i områder med der mange hyperbler skjærer hverandre, vil alle føre til hver sin estimerte posisjon. Figur 6.13 viser estimerte posisjoner på bakgrunn av de initielle posisjonene (estimatoren beskrives senere i avsnitt 6.2.2.3). For å vise sammenhengen mellom de initielle posisjonene og de estimerte posisjonene er det trukket en rød linje mellom dem. De røde linjene som viser «konvergensen» til estimatene, vil stort sett ligge langsetter hyperblene, ettersom hyperblene tross alt består av posisjoner som gir samme TDOA.



Figur 6.13: Sammenheng mellom initielle posisjoner og estimerte posisjoner.

I figuren er det totalt 577 initielle posisjoner, og dermed 577 estimat. Rundt halvparten av de initielle posisjonene fører til gale estimater, mens de resterende fører til estimat som ligger nært radarene. I håp om å redusere antall estimat, og forhåpentligvis få kun et estimat per radarposisjon, baseres algoritmen på en greedy-filosofi.

Figur 6.14 viser et overordnet flytskjema over algoritmen. Input til algoritmen er de initielle posisjonene og en liste over alle TDOA-er med tilhørende sensorposisjoner. Algoritmen beregner da først hvilke TDOA-er som kan komme fra den initielle posisjonen dersom en radar er plassert i denne posisjonen. Deretter beregnes en kriterieverdi for alle initielle posisjoner, slik at de initielle posisjonene kan rangeres etter hvor trolig det er at en radar ligger i nærheten. Estimatoren velger så den initielle posisjonen med størst kriterieverdi, og estimerer en posisjon på bakgrunn av de TDOA som tilhører den initielle posisjonen. TDOA-ene som benyttes til estimatet av posisjonen fjernes, slik at disse ikke kan gjenbrukes. Når posisjonen er estimert og TDOA-ene fjernet, begynner algoritmen på nytt. Algoritmen fortsetter helt til alle initielle posisjoner har kriterieverdier under en grense.



Figur 6.14: Flytskjema over hovedprinsippene i estimering av radarposisjoner.

6.2.2.1 TDOA som kan komme fra de initielle posisjonene

Starter med å finne hvilke TDOA som kan komme fra en eventuell radar i den initielle posisjonen. Geometrisk betyr dette å velge de TDOA som gir hyperbler som går gjennom eller nært den initielle posisjon. TDOA-ene som kan komme fra den initielle posisjonen finnes ved å sammenlikne TDOA-ene med den sanne TDOA-en for den initielle posisjonen. En TDOA kan komme fra en initiell posisjon dersom TDOA-en ligger innenfor pluss/minus a standardavvik fra de sanne TDOA-en, dvs. at

$$\left|\underline{z} - \underline{h}(\underline{x})\right| \leq a\sigma, \tag{6.1}$$

hvor σ er et standardavvik, a er antall standardavvik, $\underline{z}$ inneholder alle TDOA, og $\underline{h}(\underline{x})$ er den ulineære vektorfunksjonen som gir den sanne TDOA og hvor linje k bestemmes ved

$$h_k(\underline{x}) = \frac{1}{c}\left(\left\|\underline{x} - \underline{p}_{S2}^{N}(k)\right\| - \left\|\underline{x} - \underline{p}_{S1}^{N}(k)\right\|\right), \tag{6.2}$$

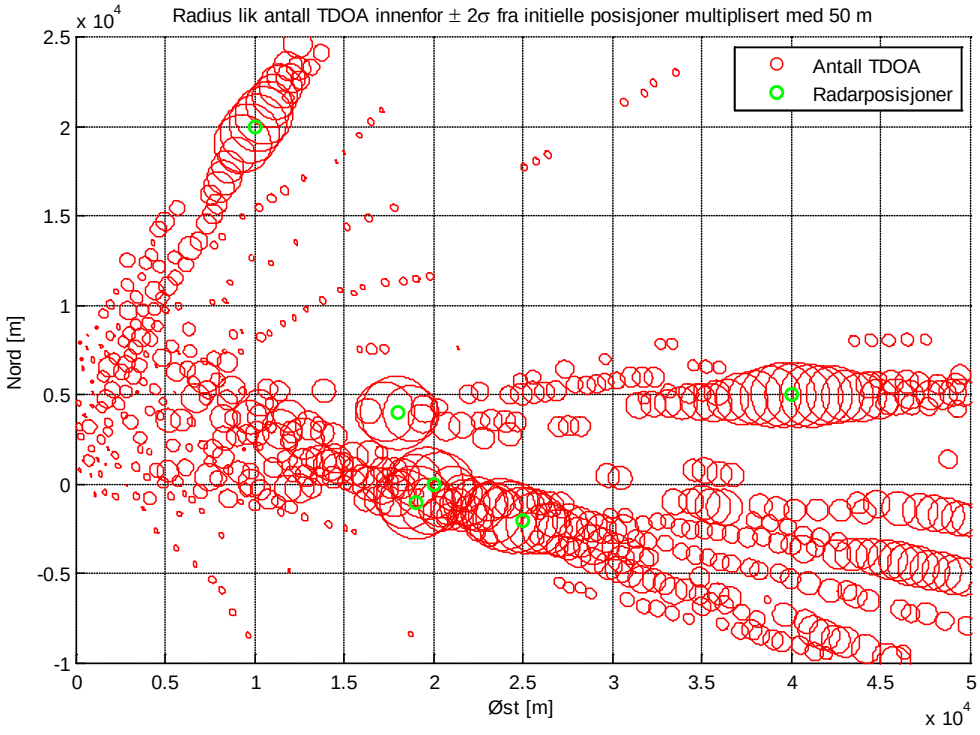
hvor $\underline{x}$ er posisjonen til den initielle posisjonen, og $\underline{p}_{S2}^{N}(k)$ og $\underline{p}_{S1}^{N}(k)$ er posisjonene til hhv. sensor 2 og sensor 1 for TDOA nummer k. I oppgaven er det valgt at differansen mellom en TDOA og den sanne TDOA må være mindre enn 2σ, hvor standardavviket gjelder for midlede TDOA (σ = 22,36 ns – antok i avsnitt 6.1.2 at hver midlede TDOA består av 10 TDOA). Spredningen til TDOA-ene som kan komme fra de initielle posisjonene beregnes ved RMS-verdien

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(z_i - h_i(\underline{x})\right)^2}, \tag{6.3}$$

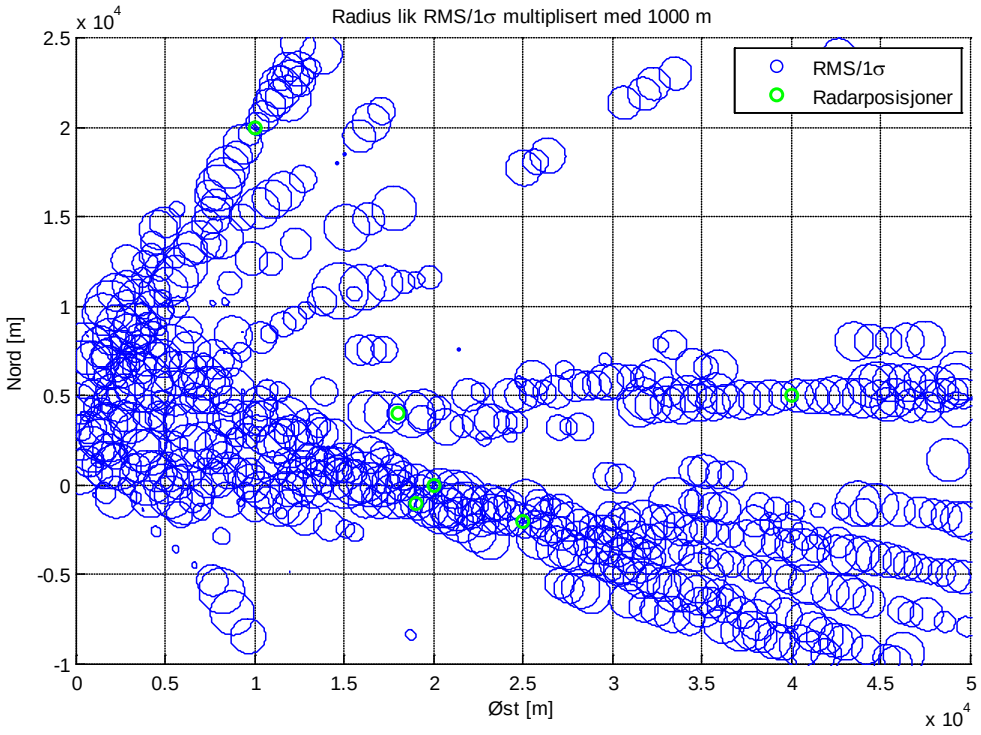
hvor z_i og $h_i(\underline{x})$ hhv. er (midlede) TDOA og sanne TDOA som tilfredsstiller formel (6.1).

Dersom den initielle posisjonen ligger nært en radar, vil svært mange TDOA kunne komme fra den initielle posisjonen, dvs. at det er mange TDOA som tilfredsstiller formel (6.1). Sannsynligvis vil sensorene motta TDOA fra radaren i alle seks måleintervallene. For initielle posisjoner langt unna radarene, vil det sannsynligvis være færre TDOA som kan komme fra den initielle posisjonen, dvs. at det er få TDOA som tilfredsstiller formel (6.1). Figur 6.15 viser antall TDOA som kan komme fra hver initielle radarposisjon, altså som tilfredsstiller formel (6.1). Figur 6.16

viser spredningen, i antall standardavvik, til TDOA-ene fra Figur 6.15. Ved å se på Figur 6.15 ses det at de røde sirklene blir større jo nærmere de ligger radarene. Det betyr at antall TDOA som kan komme fra de initielle posisjonene er høyere for initielle posisjoner nært radarene. Geometrisk betyr dette at jo nærmere radaren den initielle posisjonen ligger, jo flere hyperbler ligger i nærheten. Samtidig ses det at det er få TDOA for initielle posisjoner langt unna radarene. M.a.o. kan det ganske tydelig ses hvor nært den initielle posisjonen ligger en radar ved å se på antall TDOA som tilfredsstiller formel (6.1). Derimot ser det ut til at RMS-en i Figur 6.16 gir liten indikasjon på hvilke initielle posisjoner som ligger i nærheten av radarene. I figuren ser det ut til at initielle posisjoner nært radarene både kan ha liten RMS og stor RMS, samtidig som at initielle posisjoner langt fra radarene også kan ha liten RMS og stor RMS. Dette skylles at spredningene til TDOA-ene kan være liten selv om den initielle posisjonen ligger langt fra en radar, og spredningene til TDOA-ene kan være stor selv om den initielle posisjonen ligger nært en radar. RMS-en gir derfor ingen indikasjon på hvor nært en radar den initielle posisjonen ligger.



Figur 6.15: Antall TDOA innenfor pluss/minus to standardavvik fra sann TDOA til initielle posisjoner.



Figur 6.16: RMS/σ til TDOA innenfor pluss/minus to standardavvik fra initielle posisjoner.

6.2.2.2 Kriteriefunksjon for initielle posisjoner

Avsnitt 6.2.2.1 gir at de initielle posisjonene kan karakteriseres ut fra hvor mange TDOA som kan komme fra den initielle posisjonen. Velger derfor å bruke antall TDOA som kriteriefunksjon. Ettersom sensorene minimum må motta TDOA-er fra tre ulike sensorposisjoner (se avsnitt 5.1.1), må de initielle posisjonene minimum inneholde TDOA-er fra tre måleintervall. Kriteriefunksjonen blir da

$$J = \begin{cases} N_{TDOA}, & N_{intervall} \geq 3 \\ 0, & \text{ellers} \end{cases}, \tag{6.4}$$

hvor N_{TDOA} er antall TDOA som tilfredsttiller formel (6.1) og $N_{intervall}$ er antall måleintervall som er representert. De initielle posisjonene rangeres deretter etter sin kriterieverdi, og estimatoren estimerer en posisjon på bakgrunn av den initielle posisjonen med høyest kriterieverdi.

6.2.2.3 Estimat av radarposisjoner

Målemodellen for TDOA fra en radar i posisjonen $\underline{x}$ er gitt ved formel (5.10)

$$\underline{z} = \underline{h}(\underline{x}) + \underline{w}, \qquad \underline{w} \sim p_w(\underline{w}), \tag{6.5}$$

hvor $\underline{h}(\underline{x})$ er en ulineær vektorfunksjon av den fullstendige ukjente konstanten $\underline{x}$, og $\underline{w}$ er additiv støy med fordeling $p_w(\underline{w})$. Element k i den ulineære vektorfunksjonen er bestemt ved

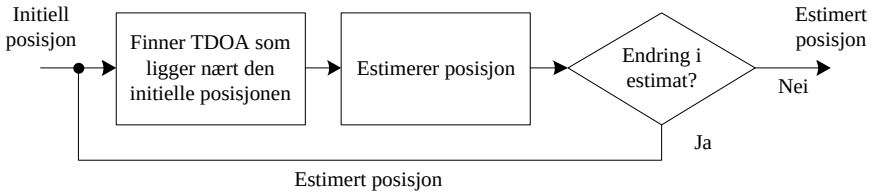
$$h_i(\underline{x}) = \frac{1}{c}\left(\left\|\underline{x} - \underline{p}_{S2}^{N}(k)\right\| - \left\|\underline{x} - \underline{p}_{S1}^{N}(k)\right\|\right). \tag{6.6}$$

hvor $\underline{p}_{S2}^{N}(k)$ og $\underline{p}_{S1}^{N}(k)$ er posisjonene til hhv. sensor 2 og sensor 1 for TDOA nr. k. Ettersom posisjonen som skal estimeres er konstant, vil målelikninga (6.5) være en Fishermodell (se appendiks D.1). Et estimat av den ukjente posisjonen $\underline{x}$ kan da beregnes ved minstekvadraters metode (se appendiks D.2). Estimatet av posisjonen vil da være den posisjon som minimaliserer kvadratavviket

$$\hat{\underline{x}} = \arg\min_{\underline{x}} \left(\underline{z} - \underline{h}(\underline{x})\right)^T \left(\underline{z} - \underline{h}(\underline{x})\right), \tag{6.7}$$

Hvor $\underline{z}$ er midlede TDOA og $\underline{h}(\underline{x})$ er den ulineære vektorfunksjonen. Estimatoren i formel (6.7) kan implementeres ved søkefunksjonen «fminsearch» i MATLAB. Funksjonen tar utgangspunkt i en initiell posisjon og søker seg frem til en $\underline{x}$ som gir et lokalt minimum.

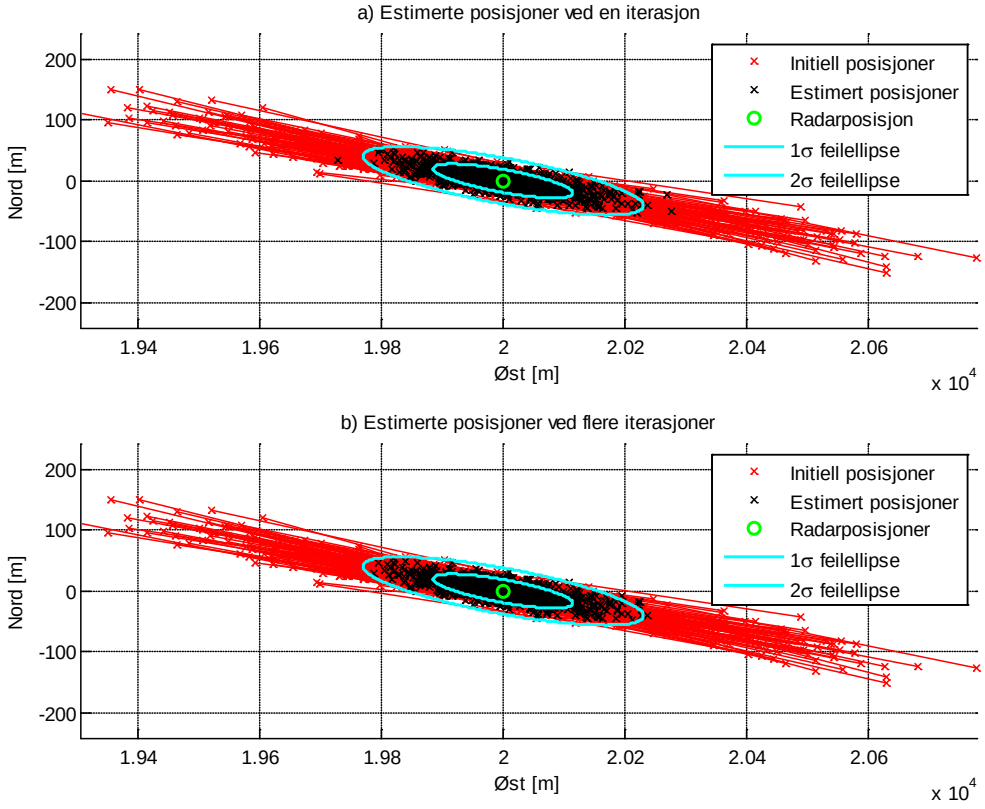
Fra avsnitt 6.2.2.1 fås et sett med initielle posisjoner og tilhørende TDOA-er, og i avsnitt 6.2.2.2 blir de initielle posisjonene rangert etter hvor trolig det ligger en radar i nærheten. Estimatoren tar da utgangspunkt i den initielle posisjonen med høyest kriterieverdi og de TDOA som kan komme fra den initielle posisjonen. Estimatoren i formel (6.7) starter da søket fra den initielle posisjonen og de TDOA som kan komme fra den initielle posisjonen, altså som tilfredsstiller formel (6.1). Deretter søker algoritmen seg frem til den posisjon som minimaliserer kvadratavviket. Det kan tenkes at den initielle posisjonen ligger langt fra den estimerte posisjonen, og at den estimerte posisjonen dermed kan bli enda bedre ved å inkludere de TDOA som kan komme fra den estimerte posisjonen. Figur 6.17 viser et flytskjema over estimeringsalgoritmen. Estimeringsalgoritmen finner da først TDOA som ligger i nærheten av den initielle posisjonen ved formel (6.1), og en posisjon estimeres ved formel (6.7). Deretter beregnes det hvilke TDOA som ligger i nærheten av den estimerte posisjonen, og en nyere og forhåpentligvis bedre posisjon estimeres. Slik fortsetter algoritmen inntil det ikke er noen forandring i den estimerte posisjonen eller et maksimum antall iterasjoner er oppnådd.



Figur 6.17: Flytskjema over estimering av radarposisjoner.

Ved å estimere en posisjon flere ganger, og hver gang ta med flere TDOA-er, kan det tenkes at den estimerte posisjonen kan «drive» bort fra en radar og at estimatet da blir dårligere. For å undersøke om estimatet blir bedre ved å estimere på en posisjon flere ganger, estimeres en posisjon ved en Monte Carlo simulering. Figur 6.18 viser to figurer; a) hvor estimatoren estimerer kun en posisjon uten noen form for å forbedre estimatet ved 1000 gjennomkjøringer av Monte Carlo simulering, og b) hvor estimatoren estimerer en posisjonen flere ganger ved 1000 gjennomkjøringer av Monte Carlo simulering. I figuren er initielle posisjoner tegnet med røde

kryss, og estimerte posisjoner med sorte kryss. Mellom de initielle posisjonene og estimerte posisjonene er det trukket en rød linje for å se hvilke initielle posisjoner og estimerte posisjoner som hører sammen. I figuren er det også tegnet opp 1σ feilellipse og 2σ feilellipse beregnet ved CRLB (se appendiks D). Estimeringsnøyaktighet og feilellipser diskuteres nærmere i kapittel 7. Av figuren ses det at å estimere posisjonen til en radar flere ganger gir svært liten forbedring, og det ligger bare 1 % flere estimat innenfor 2σ feilellipsen når radarposisjonen estimeres flere ganger kontra en gang.

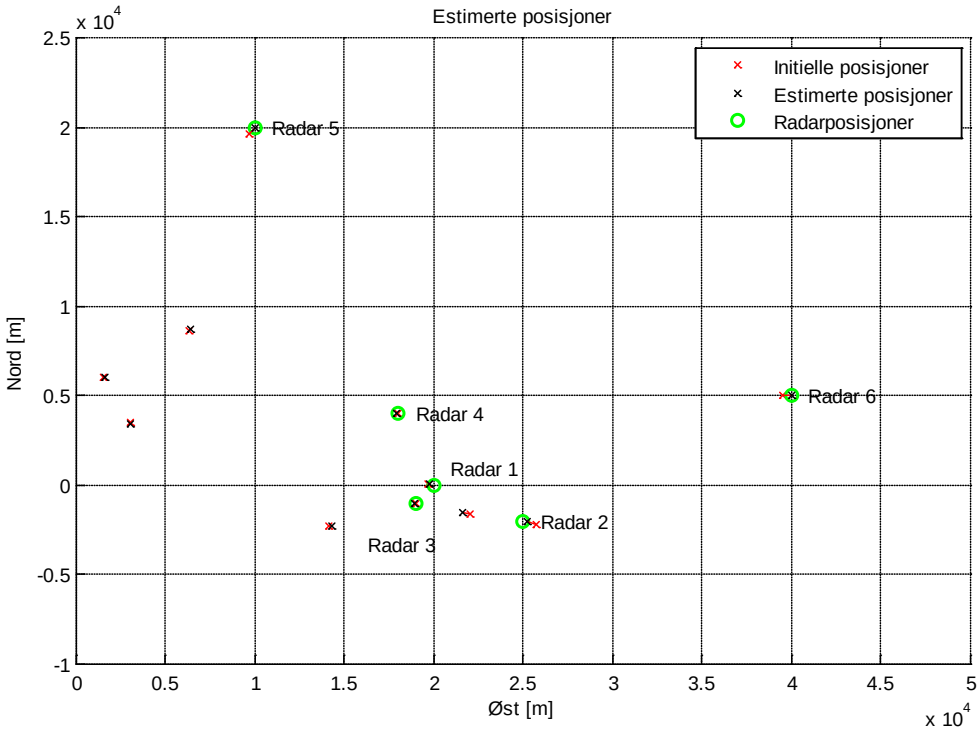


Figur 6.18: a) Estimerte posisjoner ved en iterasjon. b) Estimerte posisjoner ved flere iterasjoner.

6.2.3 Forskjeller mellom riktige og gale estimat

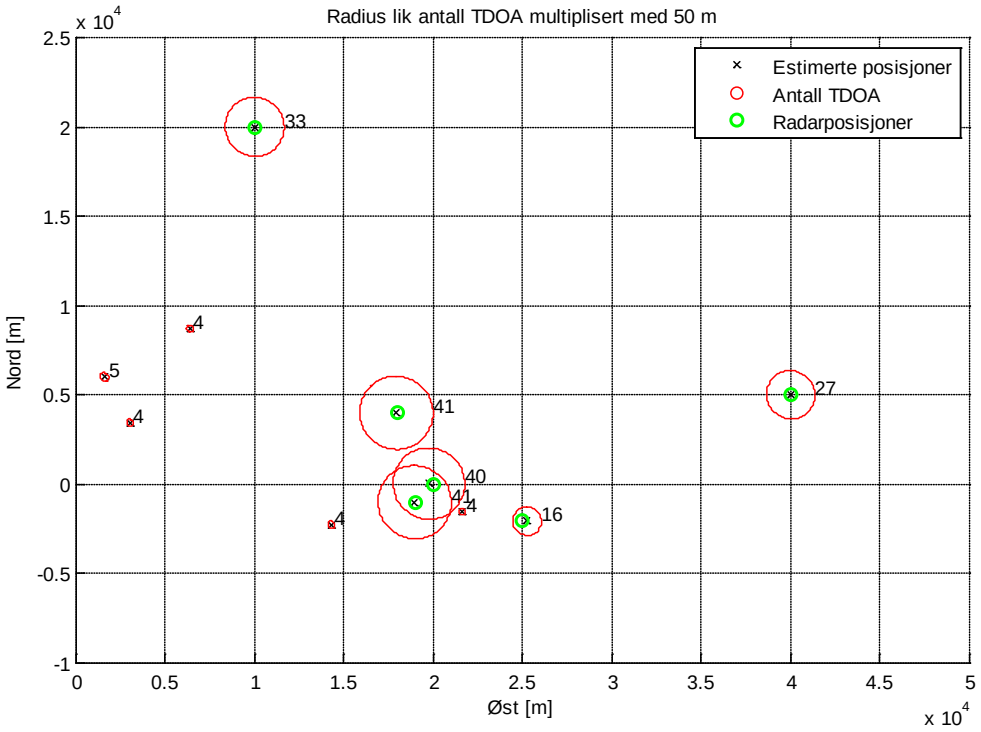
Ved å benytte metoden beskrevet i avsnitt 6.2.2, begrenses antallet estimater. Forhåpentligvis gir algoritmen et estimat for hver radarposisjon, men sannsynligvis også noen gale estimater.

Figur 6.19 viser 11 estimerte posisjoner (sorte kryss), med tilhørende initielle posisjoner (røde kryss), når sensorene mottar pulser fra seks navigasjonsradarer (grønne sirkler). I figuren har algoritmen først tatt utgangspunkt i de initielle posisjonene fra Figur 6.12, og funnet ut hvilke TDOA-er som kan komme fra hver av dem. Deretter har estimatoren tatt utgangspunkt i den initielle posisjonen med høyest kriterieverdi, og estimert en posisjon. TDOA-ene som er benyttet i estimatet er fjernet, og algoritmen har startet på nytt igjen med å finne hvilke TDOA-er som kan komme fra hver initielle posisjon. Slik har algoritmen fortsatt inntil kriterieverdiene til alle initielle posisjoner er null. I figuren er alle radarene geolokalisert ved hvert sitt estimat. I tillegg inneholder figuren også fem gale estimater som ikke tilhører noen radar.

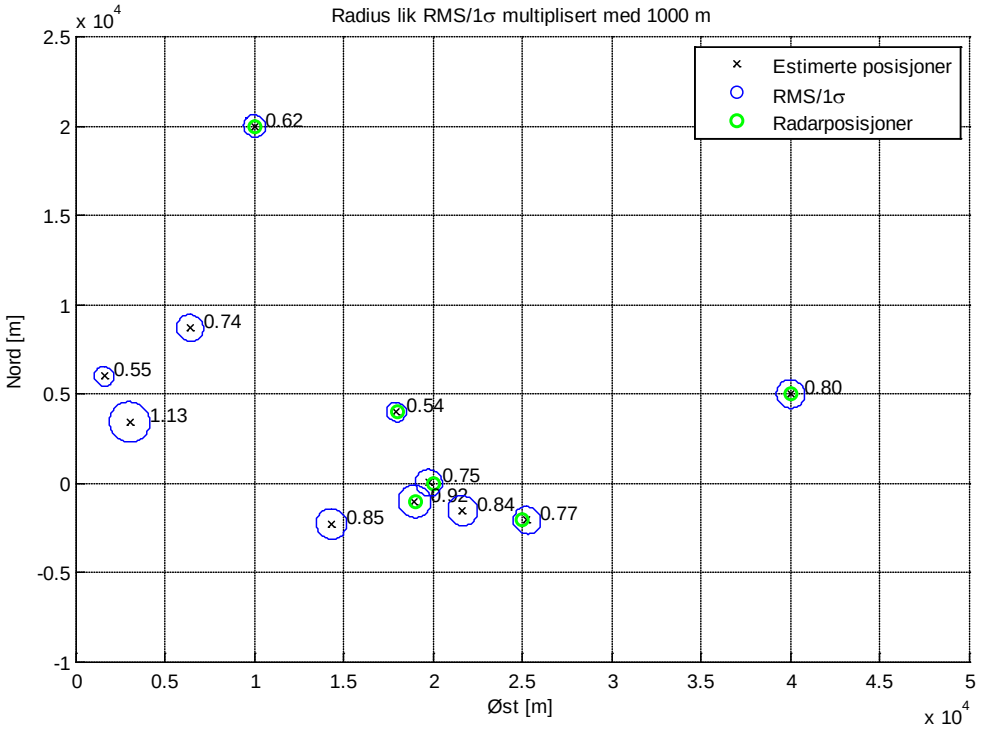


Figur 6.19: Estimerte posisjoner ved pulsassosiasjon og geolokaliseringsalgoritmen.

For å finne ut hva som skiller riktige estimat som tydelig tilhører en radar og gale estimat, ses det på antall benyttede TDOA i estimatene og spredningen til TDOA-ene. Spredningen til TDOA-ene beregnes ved formel (6.3) hvor det antas at radaren ligger i den estimerte posisjonen. Figur 6.20 viser antall TDOA og Figur 6.21 viser spredningen til TDOA-ene i antall standardavvik. Som figuren viser er antallet TDOA mye høyere for de riktige estimatene enn for de gale, mens det ikke ser ut til å være noen sammenheng mellom riktige estimat og RMS-verdien. Dette stemmer godt overens med [2], hvor det ble konkludert med at riktige og gale estimat kunne skilles ved å se på antall TDOA, mens RMS ikke gav noen informasjon om estimatet var riktig eller galt. I oppgaven benyttes det derfor kun antall TDOA til å bestemme om estimatet er riktig.



Figur 6.20: Antall TDOA som benyttes i hvert estimat.



Figur 6.21: RMS-verdien til TDOA som benyttet i hver estimerte posisjon.

7 Analyse av pulsassosiasjon og geolokaliseringsalgoritmen

I dette kapittelet undersøkes det hvor godt pulsassosiasjon og geolokaliseringsalgoritmen fra kapittel 6 fungerer på simulerte data når radarer er plassert i ulike formasjoner. Algoritmen testes både for UAS-scenarioet og satellitt-scenarioet. Analysene i kapittelet foregår ved Monte Carlo simuleringer, hvor pulsassosiasjon og geolokaliseringsalgoritmen kjøres flere ganger på hvert datasett. For hver gjennomkjøring trekkes det ny støy på ankomsttidene. Ved Monte Carlo simuleringen og den teoretisk nedre grensen for estimeringsnøyaktighet, beregnet vha. CRLB, kan estimeringsnøyaktigheten undersøkes.

Analysen er delt opp i to deler; avsnitt 7.1 inneholder analyser for UAS-scenarioet og avsnitt 7.2 inneholder analyser for satellitt-scenarioet.

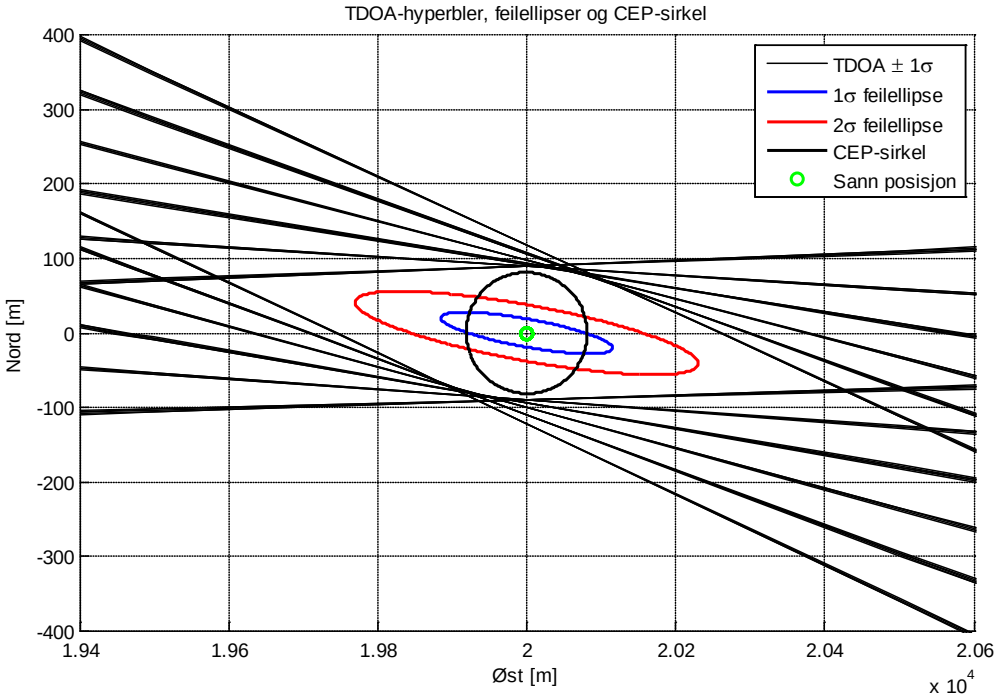
7.1 Geolokalisering av radarer i UAS-scenarioet

I analysene flyr to UAS-er nordover med en hastighet på 20 m/s, en innbyrdes avstand på 1500 m og i en høyde på 500 m. Til å generere ankomsttider til pulser fra radarer benyttes simulatoren fra kapittel 4. I simuleringen er det foretatt seks måleintervall i tidspunktene 0-10 s, 100-110 s, 200-210 s, 300-310 s, 400-410 s, og 500-510 s.

Kapittelet er delt inn i fire analyser. I avsnitt 7.1.1 undersøkes den teoretiske oppnåelige estimeringsnøyaktigheten ved CRLB. I avsnitt 7.1.2 undersøkes estimeringsnøyaktigheten til algoritmen når sensorene kun mottar pulser fra en radar og det ikke er noen tvil om hvilke pulser som hører til hvilken radar. Avsnitt 7.1.3 inneholder en analyse hvor UAS-ene mottar pulser fra seks radarer, og avsnitt 7.1.4 inneholder en analyse hvor UAS-ene mottar pulser fra 24 radarer. I analysene er radarene identiske mht. strålingsdiagram og sendereffekt, men har ulik PRI, rotasjonstid og initiell pekeretning.

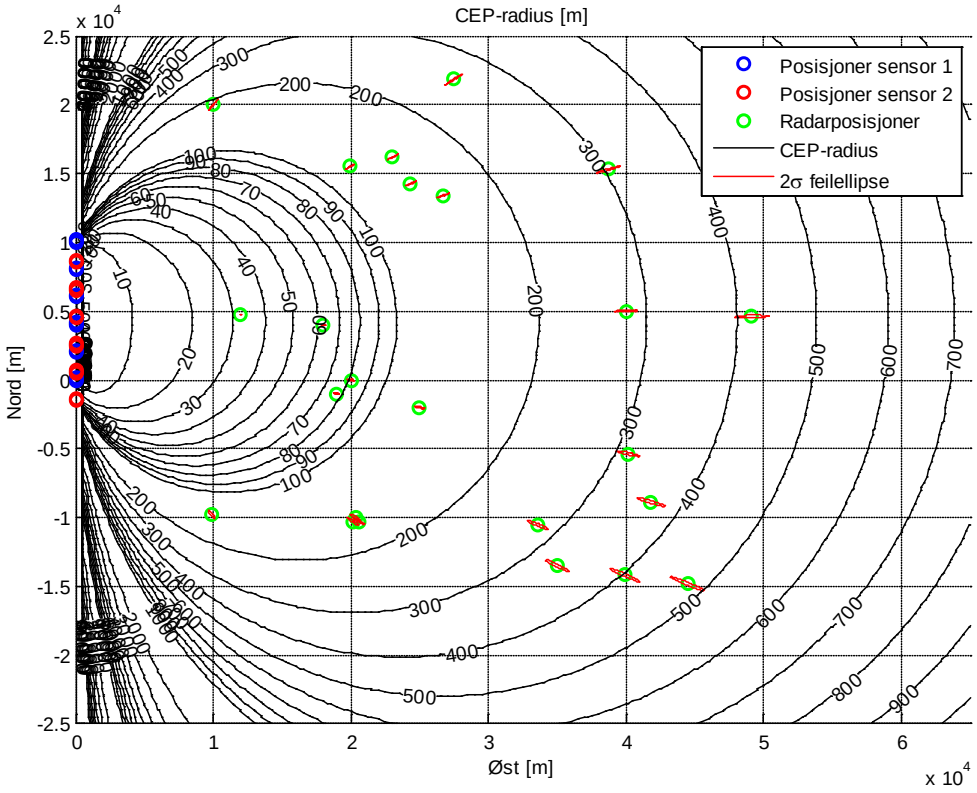
7.1.1 Analyse av oppnåelig estimeringsnøyaktighet

Estimeringsnøyaktigheten for UAS-scenarioet er avhengig av avstanden mellom radarene og UAS-ene, og hvordan radarene står i forhold til UAS-ene. Estimeringsnøyaktigheten til en radar kan analyseres vha. CRLB (se appendiks D.3). I CRLB inngår radarens posisjon og sensorenes posisjoner i måletidspunktene. I analysene om estimeringsnøyaktigheten antas det at UAS-ene mottar 27 hovedlobepasseringer (antatt rundetid på 2,2 s) fra radarene fordelt på seks måleintervall på 10 s. Antas videre at UAS-ene mottar 10 pulser per hovedlobepassering som det beregnes TDOA mellom og som midles. Av CRLB kan feilellipser og CEP-radius beregnes (se appendiks D.4 og D.5). Figur 7.1 viser estimeringsusikkerheten for en radar. Figuren inneholder 1σ og 2σ feilellipser samt CEP-sirkelen, i tillegg til hyperbler for de sanne TDOA-ene med målefeil (pluss/minus et standardavvik for midlede TDOA). Disse sanne TDOA-ene er de samme som er benyttet i CRLB. Som figuren viser ligger feilellipsene i området begrenset av hyperblene for TDOA med målefeil. Videre er ellipsene avlange, slik at nøyaktigheten er dårligere i avstand (retning øst) enn i bredde (retning nord). For å øke nøyaktigheten i avstand må vinkelen mellom hyperblene fra de ulike måleintervallene økes. Da vil det området begrenset av hyperblene bli smalere i avstand, og ellipsene blir mindre i avstand.



Figur 7.1: Hyperbler beregnet av sanne TDOA med målefeil, 1σ og 2σ feilellipser og CEP-sirkel.

Som Figur 7.1 viser er feilellipsene retningsbestemt. CEP-sirkelen derimot er uavhengig av retningen, og kan dermed benyttes til å angi estimeringsnøyaktigheten i form av konturplott. Figur 7.2 viser et konturplott med CEP-radius beregnet ved CRLB, i tillegg til 24 radarer med tilhørende 2σ feilellipser. I figuren er det også tegnet opp sensorens posisjoner i de seks måleintervallene. I beregningene av CRLB er det ikke tatt hensyn til radarenes eller UAS-enes strålingsdiagram, ettersom radaren alltid peker mot UAS-ene under hovedlobepasseringene og dipolantennes strålingsdiagram er tilnærmet omnidireksjonal for alle retninger bortsett fra flyretningen. Det antas derfor at UAS-ene vil motta 27 hovedlobepasseringer fra radarer i alle posisjoner. Av figuren ses det at CEP-radien er symmetrisk om midtpunktet til sensorene over de seks måleintervallene (omtrent i linjen som ligger i 5000 m nord). For en gitt avstand inn til sensorene, vil nøyaktigheten være best når radaren ligger på denne linjen. Dette kommer av hyperblene fra de ulike måleintervallene har størst vinkel mellom seg på denne linjen. Videre ses det i figuren at nøyaktigheten øker når avstanden inn til sensorene øker.



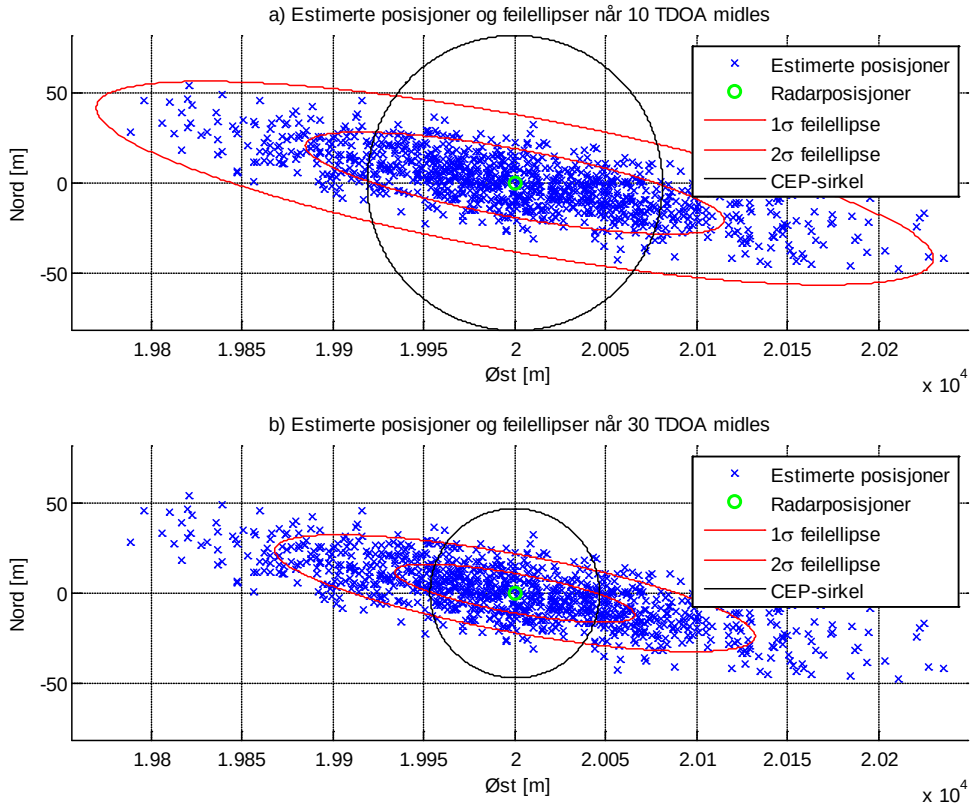
Figur 7.2: Konturplott med CEP-radius og 24 radarer med tilhørende 2σ feilellipser.

MATLAB-koden til å beregne CRLB, feilellipser og CEP-sirkel for radarposisjonene i UAS-scenarioet ligger i appendiks I.2.

7.1.2 Radarformasjon 1 – En radar

For å teste hvor godt algoritmen estimerer radarposisjoner når det ikke er noen tvil om hvilke pulser som hører sammen i de to sensorene og hvilke pulser som kommer fra samme radar, simuleres ankomsttider fra en navigasjonsradar. I simuleringen er det foretatt en Monte Carlo simulering med 1000 gjennomkjøringer. Figur 7.3 viser to figurer, hvorav begge viser estimerte posisjoner mot en radar fra Monte Carlo simuleringen. I figuren a) er det tegnet opp 1σ og 2σ feilellipser og CEP-sirkel beregnet ved CRLB (benytter samme for antakelser for CRLB som i avsnitt 7.1.1). På grunn av kort avstand mellom radar og sensorene (rundt 20 km) mottar sensorene alt mellom 10-40 pulser per hovedlobepassering som det beregnes TDOA mellom, mens det i CRLB antas 10 TDOA per hovedlobepassering (altså 10 TDOA per midlet TDOA). Dette resulterer i at CRLB gir en nedre grense for estimeringsusikkerheten som er høyere enn hva spredningen til de estimerte posisjonene er. Dette resulterer i 70,0 % av estimatene ligger innenfor 1σ feilellipsen (mot teoretisk 39,4 %), 99,4 % innenfor 2σ feilellipsen (mot teoretisk 86,5 %) og 70,7 % innenfor CEP-sirkelen (mot teoretisk 50 %). Ved å øke antallet TDOA-er per hovedlobe-passering i CRLB til 30, som er gjennomsnittet i simuleringen, bør andelen estimat innenfor de ulike feilellipsene stemme bedre overens med CRLB. Figur b) viser samme simulering som i figur a), men hvor CRLB er beregnet ut fra at 30 TDOA midles per hovedlobepassering. I figuren ligger 33,3 % innenfor 1σ feilellipsen, 79,0 % innenfor 2σ feilellipsen og 44,9 % innenfor CEP-

sirkelen. Andelen estimat innenfor ellipsene ligger litt lavere enn hva som er teoretisk oppnåelig. Grunnen et sannsynligvis at estimeringsalgoritmen kun benytter TDOA som ligger innenfor pluss/minus to standardavvik fra de sanne TDOA-ene til de initielle posisjonene. Dette fører til at estimatoren «bare» benytter 95 % av TDOA-ene, noe som igjen fører til at spredningen til estimatene blir større enn hva som er oppnåelig.

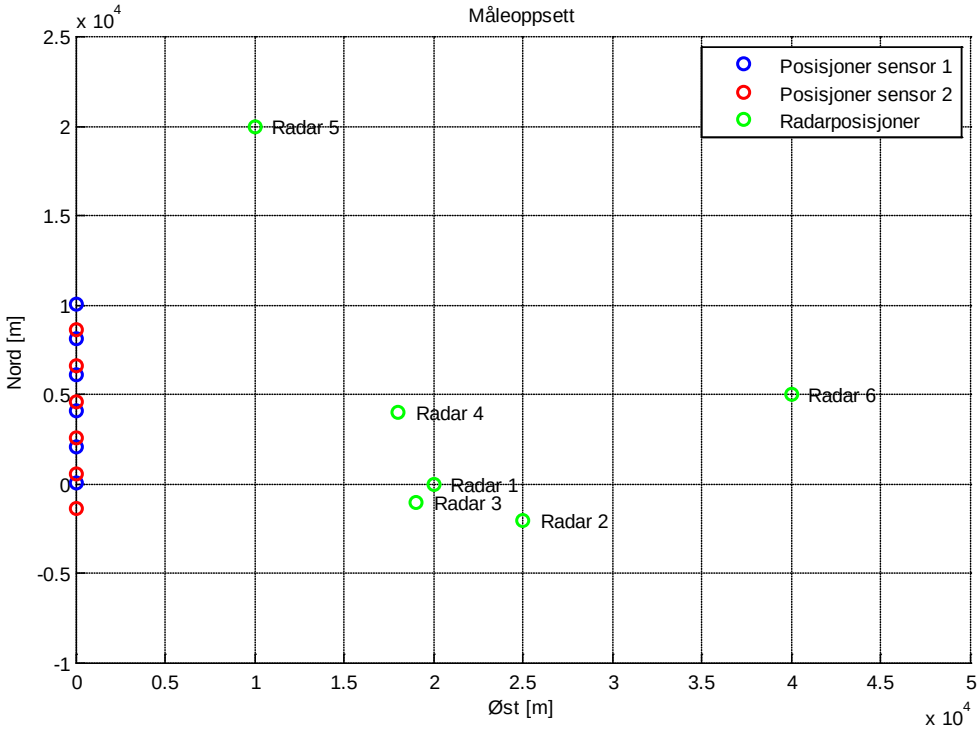


Figur 7.3: Estimerte posisjoner av en radar i UAS-scenarioet fra 1000 gjennomkjøringer av Monte Carlo simulering, 1σ og 2σ feilellipser og CEP-sirkel. I a) er 10 TDOA midlet per hovedlobepassering, og i b) 30 TDOA midlet per hovedlobepassering.

Simuleringen viser at spredningen til estimatene er noe større enn hva som er teoretisk oppnåelig. Allikevel viser simuleringen at estimatoren er forventningsrett.

7.1.3 Radarformasjon 2 – Seks radarer

Radarformasjon 2 er identisk med formasjonen benyttet under utviklingen av geolokaliseringsalgoritmen i kapittel 6, men her blir radarene geolokalisert ved 100 gjennomkjøringer av Monte Carlo simulering. Figur 7.4 viser sensorposisjonene og radarposisjonene for de seks måleintervallene som simuleres.

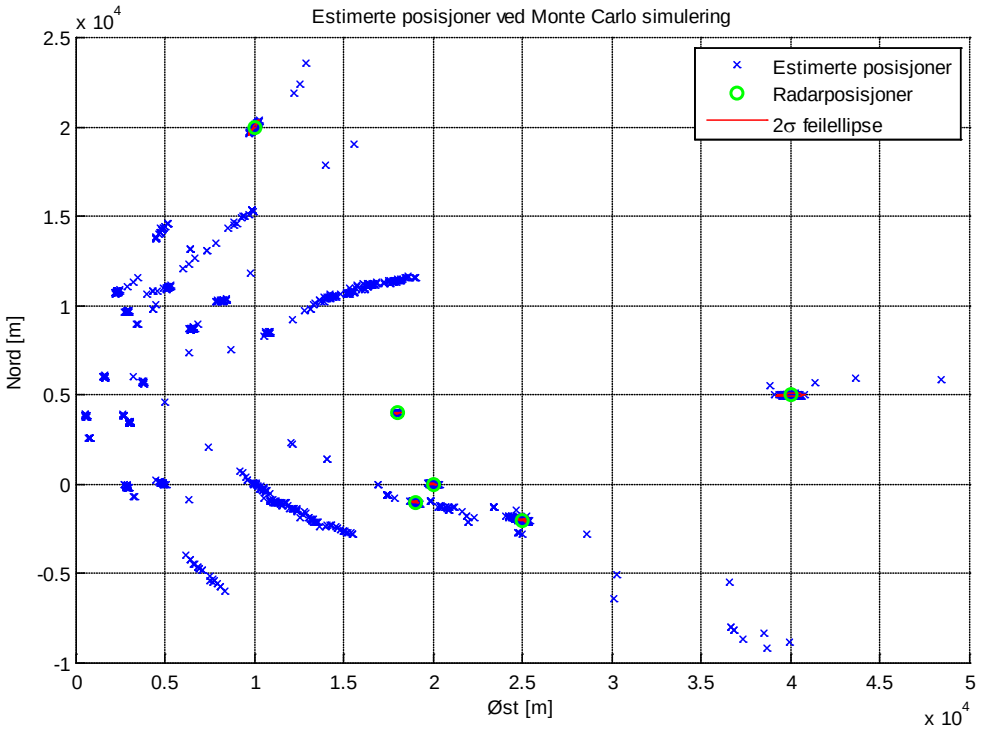


Figur 7.4: UAS-posisjoner ved seks måleintervall og seks radarer.

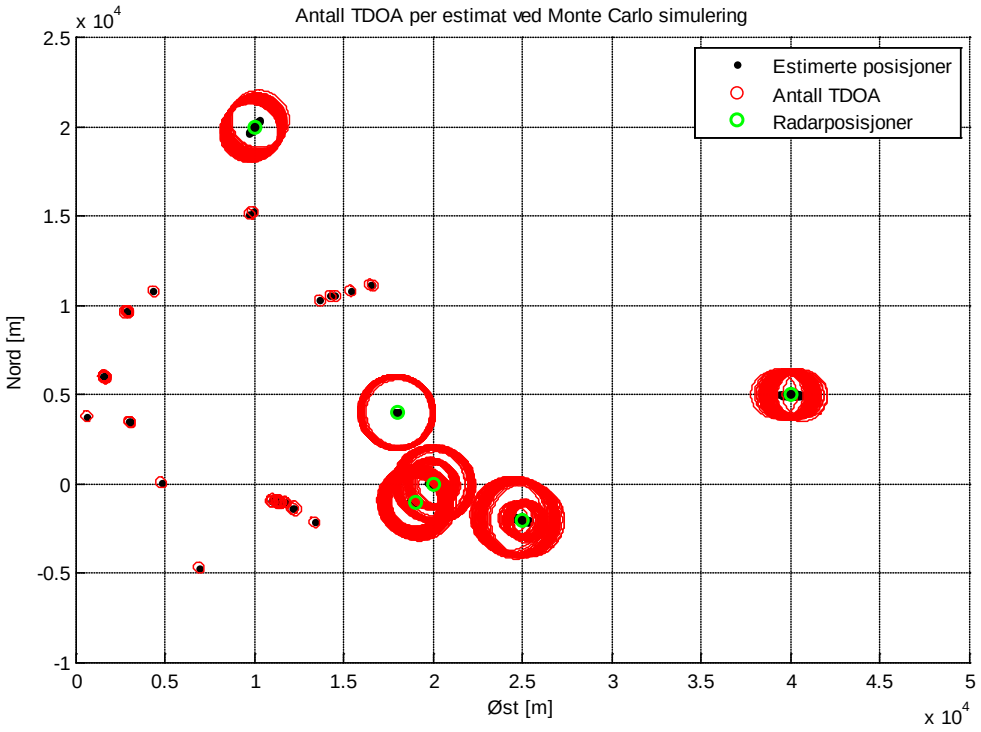
Figur 7.5 viser estimerte posisjoner ved 100 gjennomkjøringer av Monte Carlo simuleringen og Figur 7.6 viser antall TDOA som er benyttet i hvert estimat. Simuleringen gir i snitt 14 estimater for hver gjennomkjøring, hvorav de seks radarposisjonene blir estimert hver gang. I avsnitt 6.2.3 ble det begrunnet at de riktige estimatene kan skilles fra de gale estimatene ved å se på antall TDOA som benyttes i hvert estimat. I Figur 7.6 ses det at antallet TDOA er høyere for estimat tett inntil radarene enn for estimat som åpenbart er gale. Typisk inneholder de riktige estimatene opp mot fire ganger så mange TDOA som de gale estimatene. Det er verdt å merke at de gale estimatene ofte havner på de samme stedene i Monte Carlo simuleringen. Dette ses ved «linjer» med estimat i Figur 7.5. Dette skylles at hyperblene vil ligge i de samme områdene, selv ved ulikt trukket støy, og at estimatene da vil ligge i disse områdene. Tabell 7.1 inneholder andelen av estimatene som ligger innenfor de ulike feilellipsene. Andelen estimat for de ulike radarene, bortsett fra radar 2, som ligger innenfor feilellipsene er høyere enn hva som er teoretisk oppnåelig ifølge CRLB. Som nevnt i avsnitt 7.1.2 skylles dette at det i simuleringene midles 10-40 TDOA-er per hovedlobepassering, mens det i CRLB antas at 10 TDOA midles.

Tabell 7.1: Geolokaliseringsnøyaktighet ved 100 gjennomkjøringer av Monte Carlo simulering.

Radar	1σ feilellipse	2σ feilellipse	CEP-sirkel
1	45 %	89 %	45 %
2	15 %	43 %	30 %
3	54 %	92 %	66 %
4	56 %	97 %	75 %
5	60 %	100 %	68 %
6	56 %	92 %	68 %

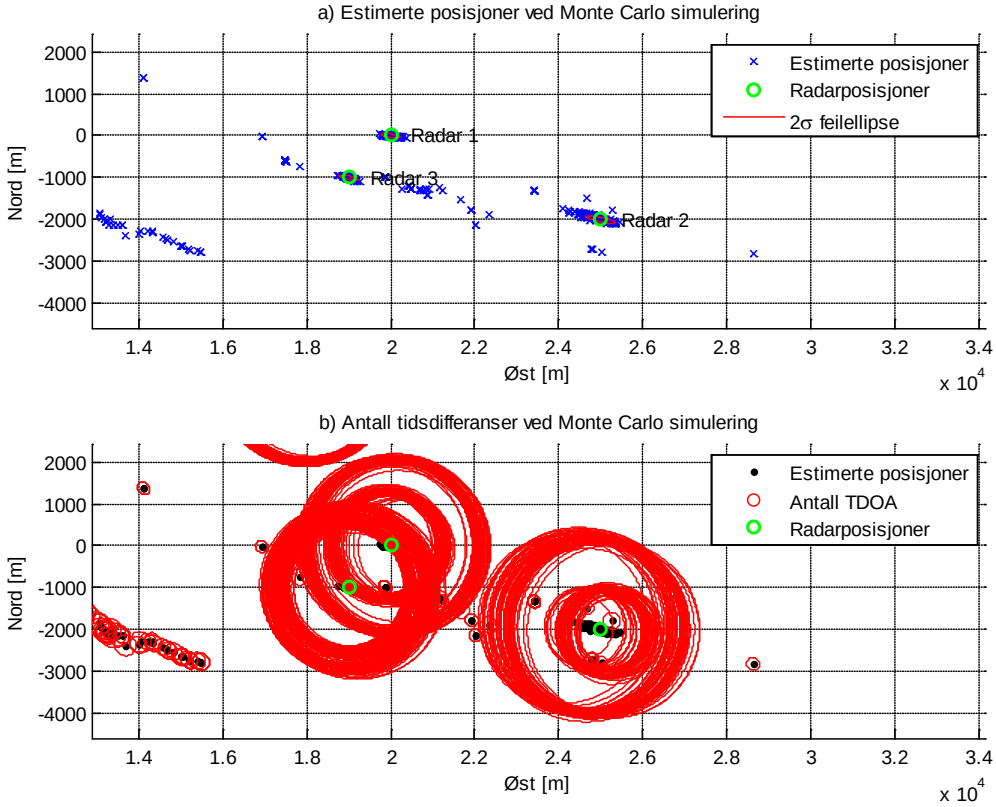


Figur 7.5: Estimerte posisjoner fra 100 gjennomkjøringer av Monte Carlo simulering, 2σ feilellipser og seks radarposisjoner.



Figur 7.6: Antall TDOA benyttet i hvert estimat fra 100 gjennomkjøringer av Monte Carlo simulering (radius lik antall TDOA multiplisert med 50 m).

Figur 7.7 viser et utsnitt av Figur 7.5 og Figur 7.6. Figur a) viser estimerte posisjoner, om figur b) antall TDOA benyttet i hvert estimat. Algoritmen gir i hver gjennomkjøring av Monte Carlo simuleringen estimater som ligger i nærheten av de tre radarene. Ved å se på antallet TDOA skiller disse estimatene seg tydelig fra de gale estimatene. I figur b) har de røde sirklene (antall TDOA) for radar 1 og 2 to ulike radier. Disse to radarene ligger på linje, sett fra sensorene. Dette fører til at mange av TDOA-ene fra radar 1 også kan tilhøre radar 2, TDOA-ene fra radar 2 kan også tilhøre radar 1. Geometrisk vil det si at hyperblene til radar 1 også går gjennom eller nært radar 2, og vice versa. I estimeringsalgoritmen velger estimatoren den initielle posisjonen men flest TDOA-er. Etter at posisjonen er estimert fjernes de benyttede TDOA-ene, slik at estimatoren ikke kan benytte disse i andre estimerte posisjoner. Dette fører til at dersom radar 1 estimeres først, vil radar 2 miste mange av sine TDOA-er. Tilsvarende gjelder dersom radar 2 estimeres først, så vil radar 1 miste mange av sine TDOA-er. På grunn av at det trekkes ulik støy for hver Monte Carlo kjøring, vil estimatoren velge ulike initielle posisjoner fra gjennomkjøring til gjennomkjøring. Dette ses tydelig ved at både radar 1 og radar 2 annenhver gang har mange TDOA-er og få TDOA-er. I figur a) ses det at spredningen til estimatene for radar 2 er større enn for de andre radarene. Dette vises også i Tabell 7.1 ved at andelen estimater innenfor de ulike feilellipsene er lavere enn for de andre radarene. I figur b) ser det ut til at mange av estimatene lengst unna radar 2 har flest TDOA-er. Det betyr at de estimatene av radar 2 som ligger lengst unna den radarposisjonen, er de estimatene hvor estimatoren tar med TDOA-er fra radar 1. Dette betyr at estimatet av radar 2 blir dårligere når estimatoren tar med TDOA-er som tilhører andre radarer.

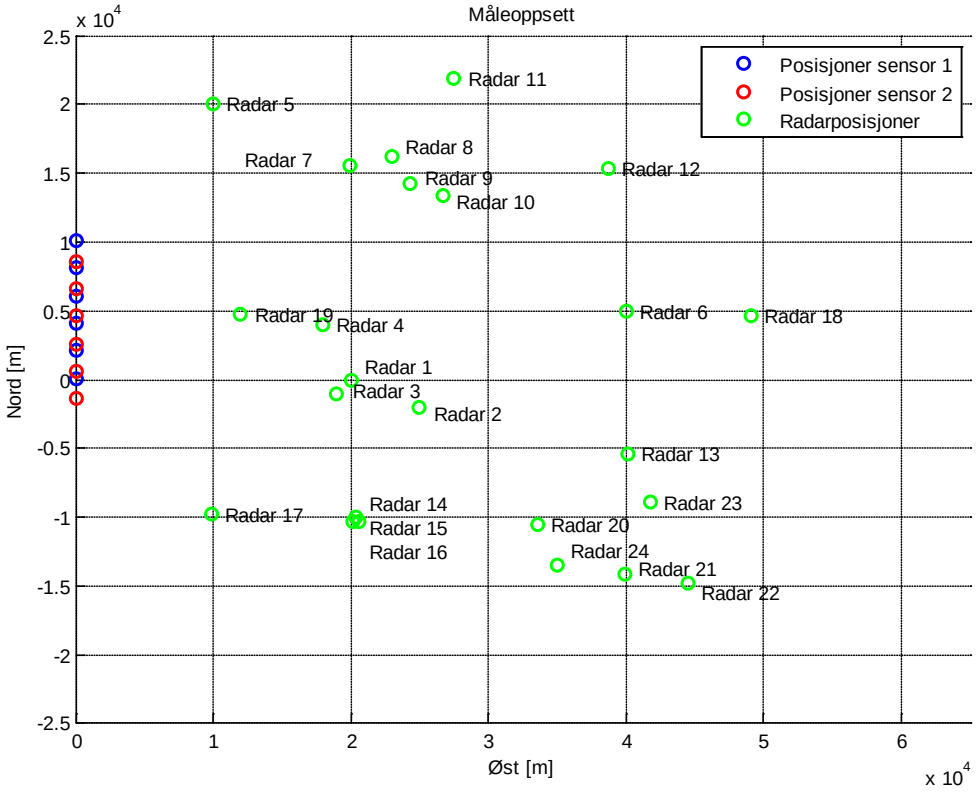


Figur 7.7: a) Estimerte posisjoner fra 100 gjennomkjøringer av Monte Carlo simulering, 2σ feilellipser og seks radarposisjoner. b) Antall TDOA benyttet i hvert estimat fra 100 gjennomkjøringer av Monte Carlo simulering (radius lik antall TDOA multiplisert med 50 m).

Av Monte Carlo simuleringen på radarformasjon 2 ses det at algoritmen estimerer alle radarene for hver gjennomkjøring, og at det er greit å skille de riktige estimatene fra de gale estimatene basert på antall TDOA i hvert estimat. Allikevel kan det bli et potensielt problem ved algoritmevalget når to radarer ligger slik at TDOA-ene fra den ene radaren også kan komme fra den andre radaren.

7.1.4 Radarformasjon 3 – 24 radarer

Radaroppsett 3 inneholder 24 radarer, hvorav radar 1 til 6 er like som for radarformasjon 2. I analysen ses det først på en simulering og deretter Monte Carlo simulering med 100 gjennomkjøringer. Figur 7.8 viser sensorposisjonene og radarposisjonene for de seks måleintervallene. I simuleringen er de fleste radarene spred tilfeldig utover, mens radar 6 og 18, radar 14, 15 og 16 og radar 20, 21, og 22 er plassert slik at algoritmen trolig får problemer med å estimere posisjoner til hver enkelt av dem.

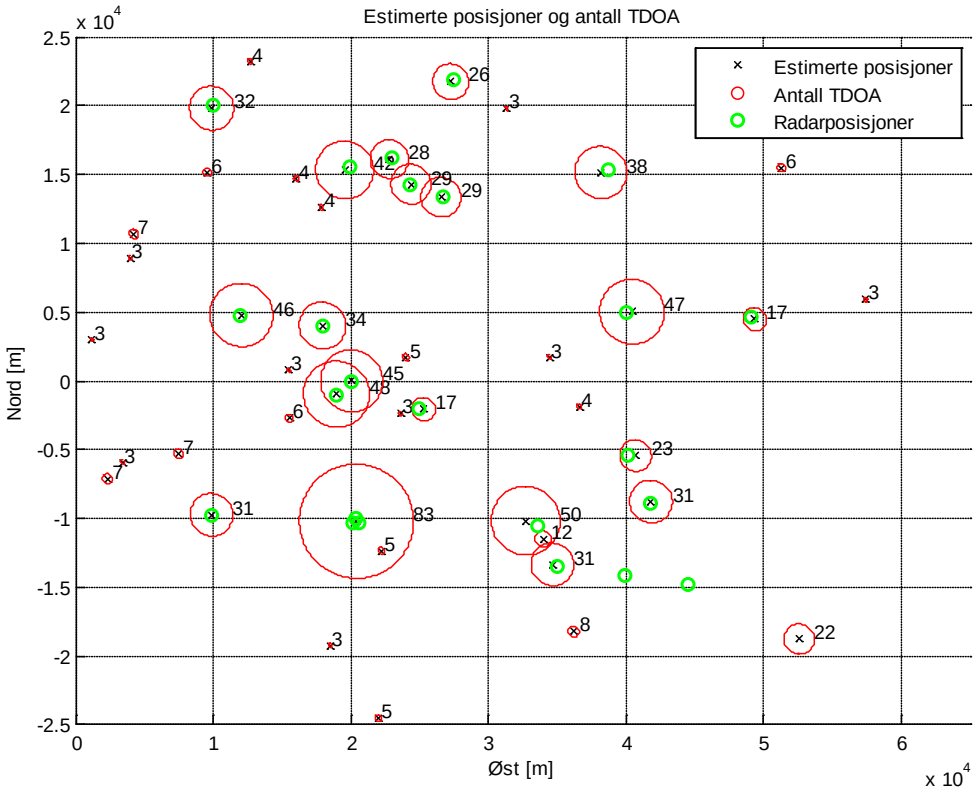


Figur 7.8: UAS-posisjoner ved seks måleintervall og 24 radarer.

7.1.4.1 En simulering

Figur 7.9 viser 45 estimerte posisjoner når sensorene mottar pulser fra 24 radarer og antall TDOA som er benyttet i hvert estimat. Det forventes å motta rundt 20-30 hovedlobepasseringer i løpet av simuleringen (gitt en rotasjonstid på 2,2 s) som er utført, noe som ideelt gir 20-30 midlede TDOA. Dette er informasjon som ikke skal brukes i oppgaven, radarene kun skal geolokaliseres ved å benytte TDOA. Allikevel ses det at estimatene som ligger nært radarposisjonene har rundt 20 eller

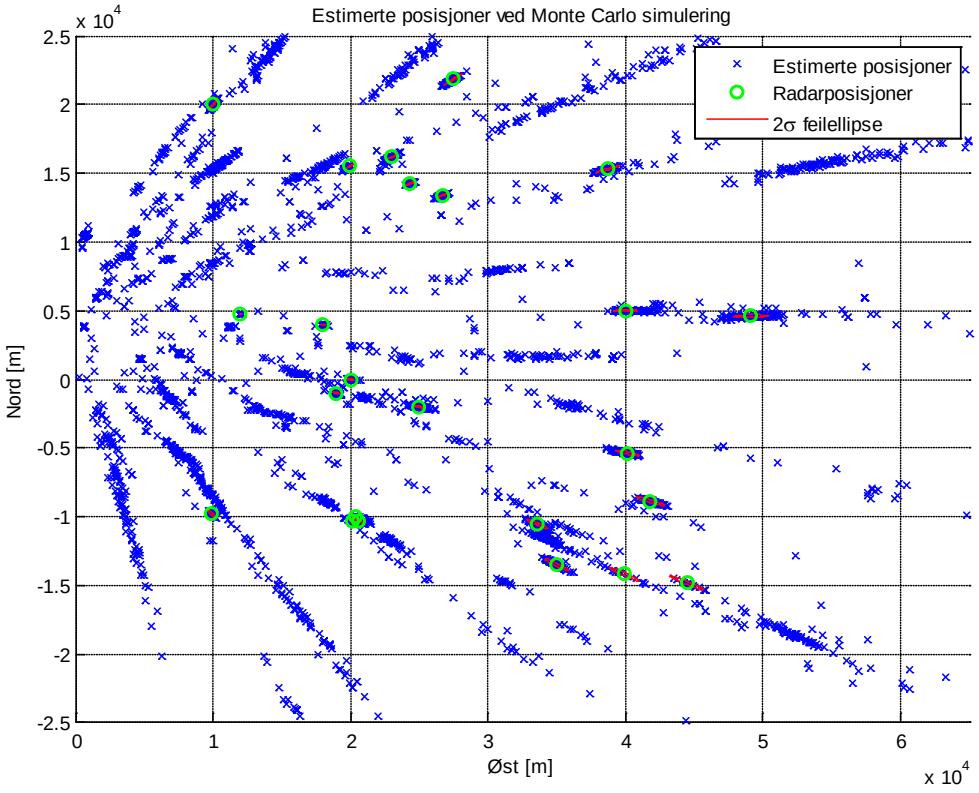
flere TDOA-er per estimat, mens de gale estimatene typisk har færre enn 10 TDOA-er. I simuleringen er det to situasjoner å ta tak i; radar 14, 15 og 16, og radar 20, 21, 22. Radar 14, 15 og 16 ligger svært tett, og algoritmen gir kun et estimat av de tre radarene. Av figuren ses det at det inngår 83 TDOA i dette estimatet. Dette stemmer godt med at antall TDOA er det tredoble av antall forventede TDOA. Radar 20, 21 og 22 ligger på linje, hvorav de to sistnevnte ligger i skyggen til radar 20. I simuleringen har estimatet av radar 20 hele 50 TDOA, noe som tyder på at den stjeler TDOA-er fra radar 21 og 22. Samtidig gir TDOA-ene som tilhører radar 21 og 22, mens om ikke er benyttet i estimatet av radar 20, et estimat i posisjonen (-19 km nord, 53 km øst) med et høyt antall TDOA.



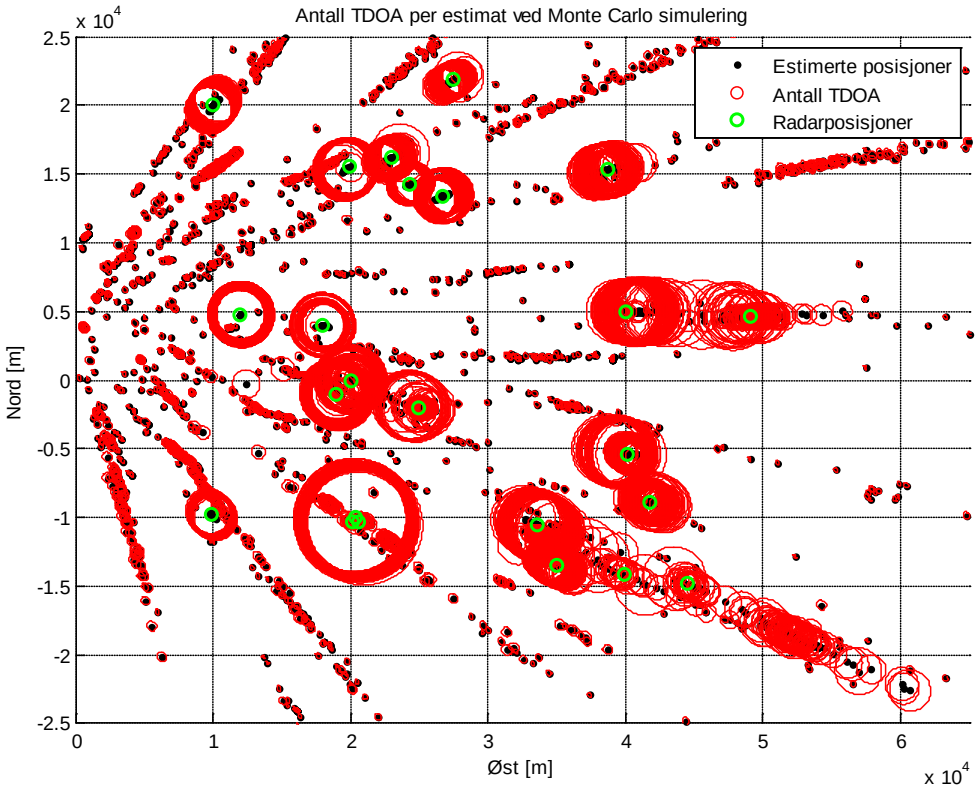
Figur 7.9: Estimerte posisjoner fra en simulering, 24 radarposisjoner og antall TDOA benyttet i hvert estimat (radius lik antall TDOA multiplisert med 50 m).

7.1.4.2 Monte Carlo simulering

Figur 7.10 viser estimerte posisjoner ved 100 Monte Carlo gjennomkjøringer og Figur 7.11 viser antall TDOA som er benyttet i hvert estimat. Simuleringen gir i snitt 45 estimat per Monte Carlo gjennomkjøring, hvorav 19 av radarposisjonene estimeres i hver gjennomkjøring. Ved å se på antall TDOA per estimat, ser det ut til at de riktige estimatene skiller seg fra de gale estimatene ved at antall TDOA er mye høyere. Typisk inneholder de riktige estimatene alt fra 17 TDOA og oppover, mens de gale estimatene ligger rundt 4-5 TDOA. I simuleringen klarer ikke algoritmen å skille radar 14, 15 og 16. Dette skylles at radarene ligger så tett at alle TDOA-ene som tilhører en av radarene også kan tilhøre de to andre radarene. Algoritmen gir derfor et estimat for de tre radarene. Antall TDOA for disse tre radarene ligger rundt 80, og er tre ganger så høyt som det forventes å motta.

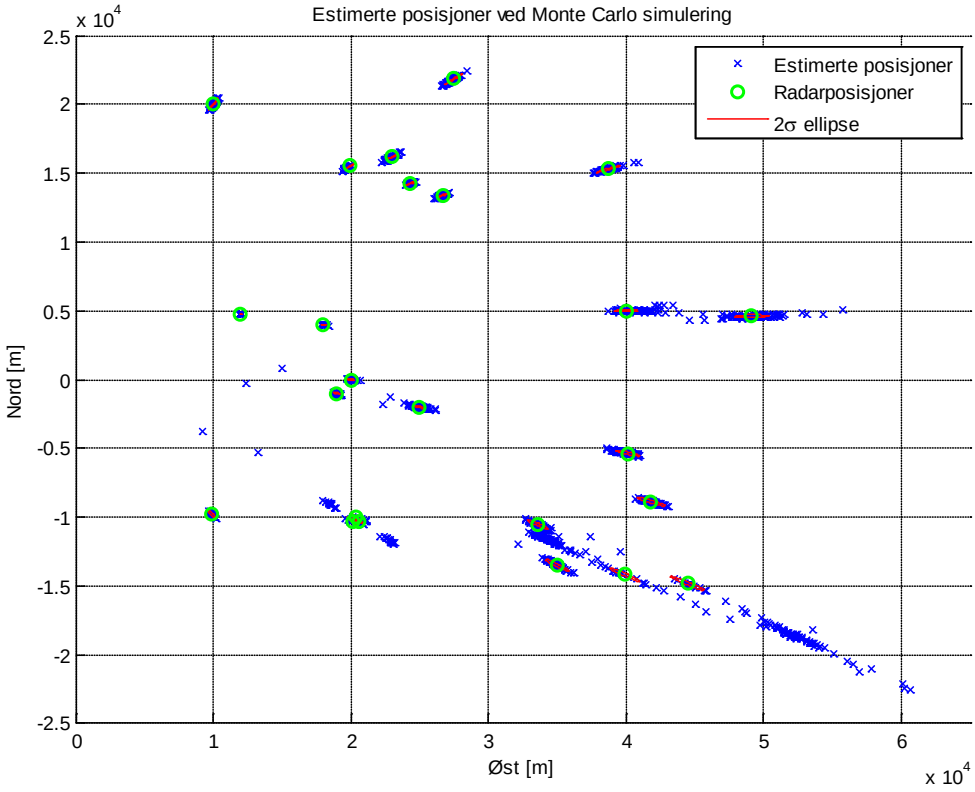


Figur 7.10: Estimerte posisjoner fra 100 gjennomkjøringer av Monte Carlo simulering, 2σ feilellipser og 24 radarposisjoner.



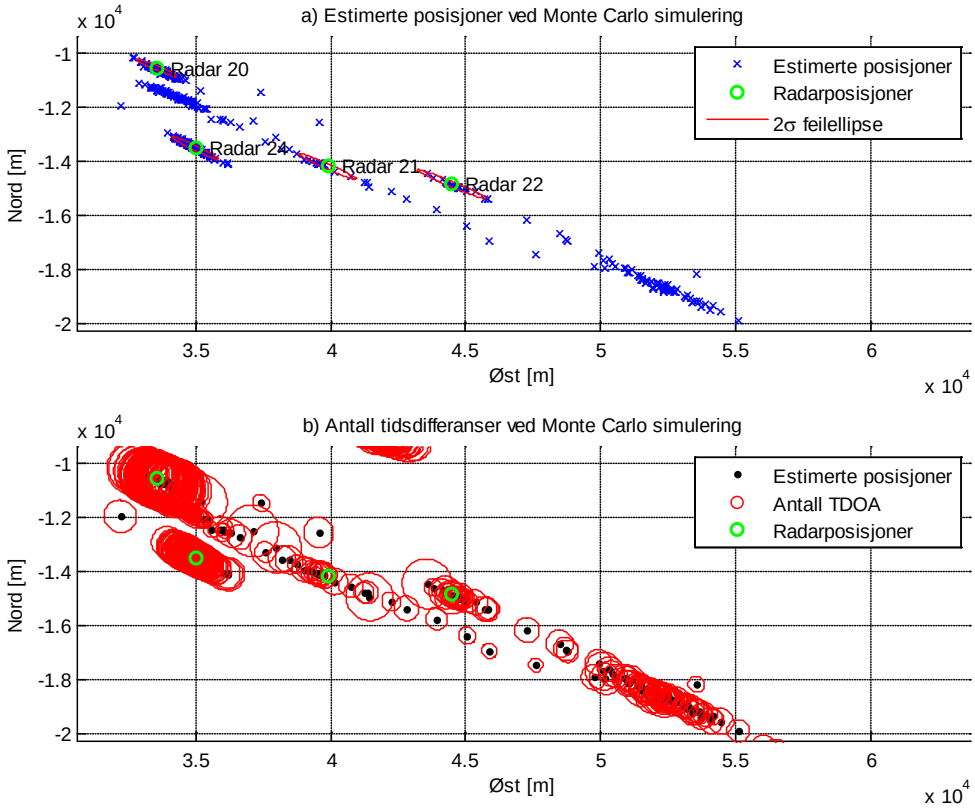
Figur 7.11: Antall TDOA benyttet i hvert estimat fra 100 gjennomkjøringer av Monte Carlo simulering (radius lik antall TDOA multiplisert med 50 m).

I Figur 7.11 er det forholdsvis greit å skille de riktige estimatene fra de gale estimatene ved å se på antall TDOA som er benyttet i hvert estimat. Ved å terskle estimatene etter hvor mange TDOA som benyttes bør antall estimat reduseres. I Figur 7.9 inneholder de fleste gale estimatene mindre enn 10 TDOA. Figur 7.12 viser samme simulering som Figur 7.10, men hvor estimatene er tersklet og må inneholder mer enn 10 TDOA-er. Svært mange gale estimater forsvinner, men det er fortsatt noen typiske områder hvor Monte Carlo simuleringen viser at algoritmen gjentatte ganger gir gale estimater. Dette gjelder spesielt for radar 20, 21 og 22, men også områdene rundt radar 14, 15 og 16, og mellom radar 6 og 18.



Figur 7.12: Tersklede estimerte posisjoner fra 100 gjennomkjøringer av Monte Carlo simulering, 2σ feilellipser og 24 radarposisjoner.

Figur 7.13 viser to figurer hvorav begge viser et utsnitt av radar 20, 21 og 22 fra Figur 7.10 og Figur 7.11. I figuren vises det kun estimater som inneholder med enn 10 TDOA-er. Figur a) viser estimerte posisjoner og figur b) viser antall TDOA benyttet i estimatene. I figuren ses det tydelig at radar 20 har mange flere estimater enn radar 21 og 22, samtidig som det ligger mange gale estimater i området (-18 til -20 km nord, 50 til 55 km øst). Videre er antallet TDOA som benyttes for de gale estimatene høyt, og kan lett blandes sammen med de riktige estimatene.



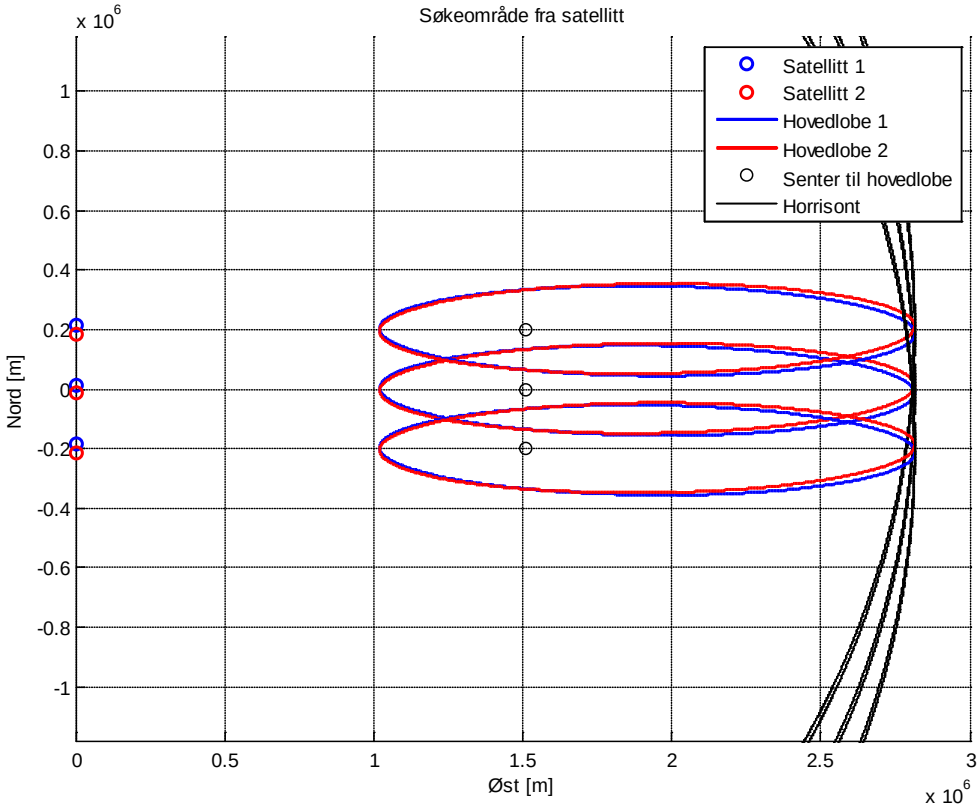
Figur 7.13: a) Tersklede estimerte posisjoner fra 100 gjennomkjøringer av Monte Carlo simulering, 2σ feilellipser og radarposisjoner for radar 20, 21 og 22. b) Antall TDOA benyttet i hvert estimat fra 100 gjennomkjøringer av Monte Carlo simulering (radius lik antall TDOA multiplisert med 20 m).

Monte Carlo simuleringen viser at algoritmen stort sett estimerer posisjonen til radarene når sensorene mottar pulser fra 24 radarer. Dette viser at algoritmen er i stand til å assosiere pulser i de to sensorene og assosiere TDOA-er fra samme radarer i et tilfelle hvor signalmiljøet er komplisert. Allikevel er det et par situasjoner hvor algoritmen ikke fungerer godt. For de tre radarene 14, 15 og 16 gir algoritmen et estimat, mens for radar 20, 21 og 22 dannes det gale estimat med svært mange TDOA-er og som dermed kan tolkes som riktige estimater. I disse to situasjonene må det andre metoder til for å estimere posisjonene til radarene..

7.2 Geolokalisering av radarer satellitt-scenarioet

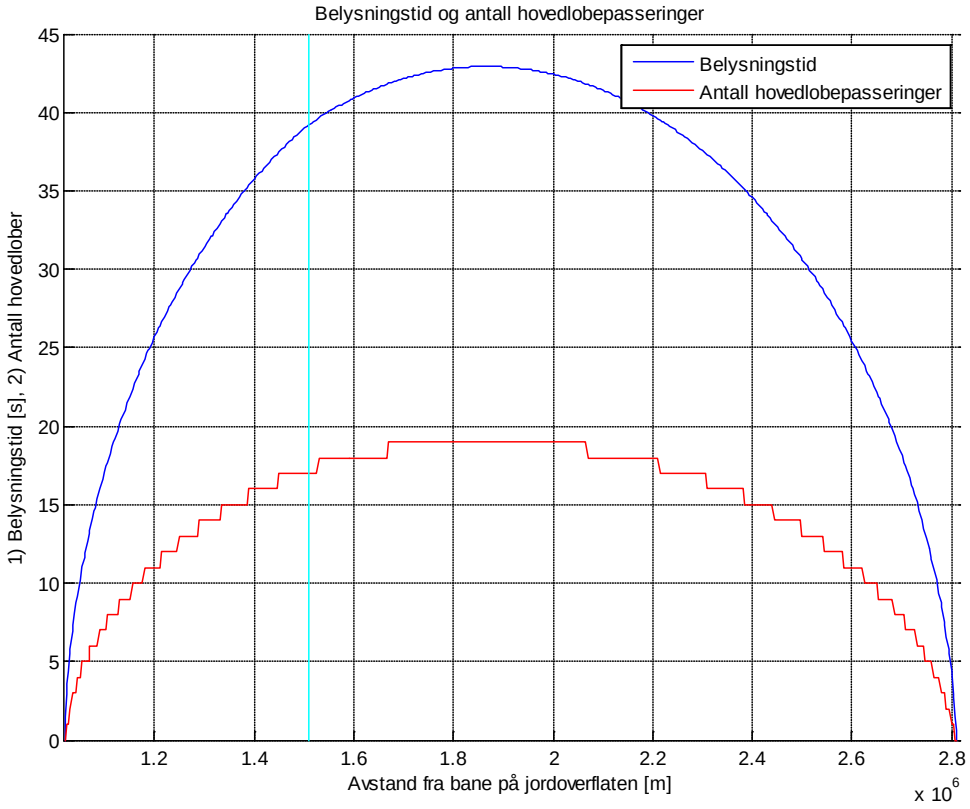
I satellitt-scenarioet ligger begge satellittene i en lav polarbane. Etter å ha introdusert en flat jord i avsnitt 3.2.1, ligger begge satellittene i en bane 290 km over jordoverflaten. Satellittenes hastighet er 6854 m/s og avstanden mellom satellittene er 30 km. Satellittene er rotert slik at deres hovedlober dekker størst mulig område nede på jordoverflaten. Lar begge satellittene peke på et punkt midt mellom satellittene i en avstand på 1508,4 km fra satellittbanen nede på jordoverflaten. Da vil den felles hovedloben nede på jordoverflaten, eller det opplyste området, dekke området mellom 1019,2 km og 2814,4 km nede på jordoverflaten, hvor sistnevnte avstand tilsvarer jordkrumningen for den kuleformede jorden. Til forskjell fra UAS-scenarioet simuleres kun hele overflyvninger i satellitt-scenarioet. Figur 7.14 viser to satellitter med hver sin hovedlobe nede på den flate jorden ved tre tidspunkter. I simuleringen har begge satellittene hovedlober med en lobebredde på 10° i både asimut og elevasjon. I figuren er det også tegnet opp

horisonten (sorte linjer) som tilsvarer jordkrumningen for den kuleformede jorden. Ettersom satellittene ikke vil kunne motta pulser fra radarer som ligger bak jordkrumningen på den kuleformede jorden, vil horisonten ses på som maksimumsavstanden ut til radarene.



Figur 7.14: Satellitt-posisjoner og deres hovedlober nede på jordoverflaten ved tre. I en avstand på 2814,4 km fra satellittene ligger horisonten for den kuleformede jorden.

På grunn av geometrien til satellittenes hovedlober, vil radarer i ulike avstander fra satellittbanen ha ulik tid i det opplyste området. Figur 7.15 viser tiden radarer, ved ulik avstand fra satellittbanen, er innenfor det opplyste området. I figuren vises også et forventet antall hovedlobepasseringer (ved en rotasjonstid på 2,2 s). Som figuren viser vil satellittene motta færre hovedlober fra radarer som ligger nært endepunktene til det opplyste området, enn radarer som ligger midt i det opplyste området.



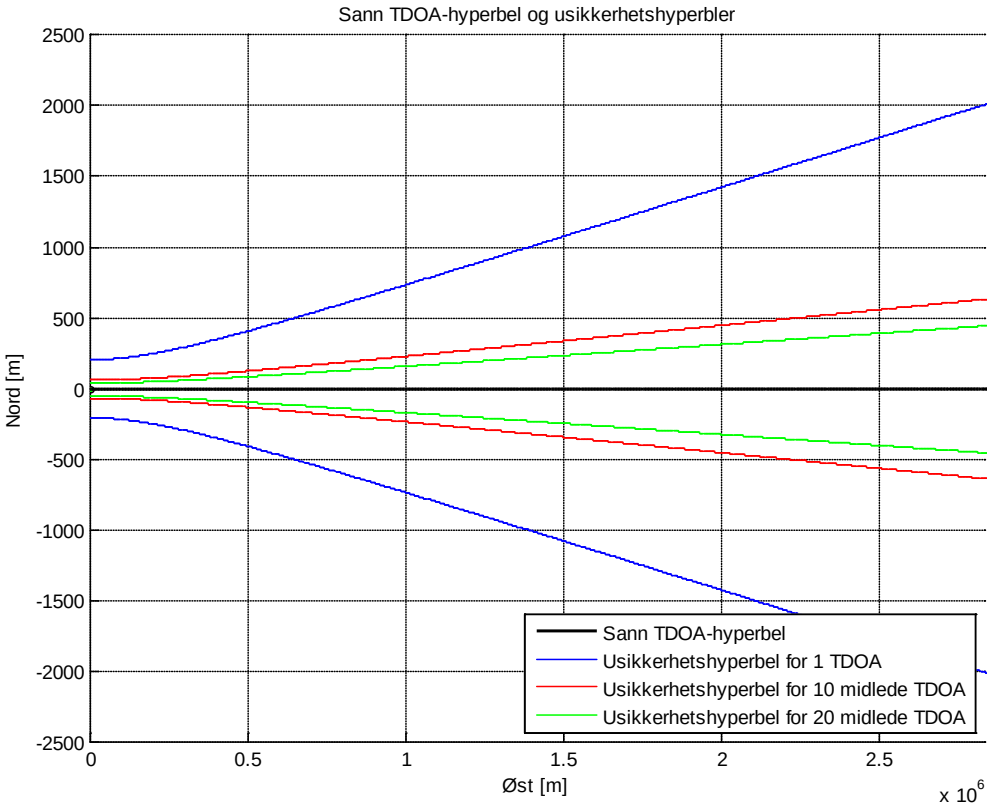
Figur 7.15: Tiden en radar vil være innenfor det opplyste området i løpet av en hel overflyvning (blå kurve) og antall hovedlobepasseringer i løpet av den samme tiden (rød kurve). Vertikal lys blå linje viser senter av satellittenes hovedlober.

Avsnittet er delt inn i fem analyser. I avsnitt 7.2.1 undersøkes den teoretiske oppnåelige estimeringsnøyaktighet ved CRLB. I avsnitt 7.2.2 undersøkes estimeringsnøyaktigheten når satellittene mottar pulser fra en radar og hvor det ikke er noen tvil om hvilke pulser som hører sammen. Avsnitt 7.2.3 inneholder en analyse hvor satellittene mottar pulser fra 10 radarer spredt utover hele det opplyste området, mens i avsnitt 7.2.4 mottar satellittene pulser fra 10 radarer tett plassert i senter av det opplyste området. I avsnitt 7.2.5 er antall radarer økt til 50 for å teste algoritmen når satellittene mottar pulser fra mange radarer. I analysene er radarene identiske mht. strålingsdiagram og sendereffekt, men har ulik PRI, rotasjonstid og initiell pekeretning.

7.2.1 Analyse av oppnåelig estimeringsnøyaktighet

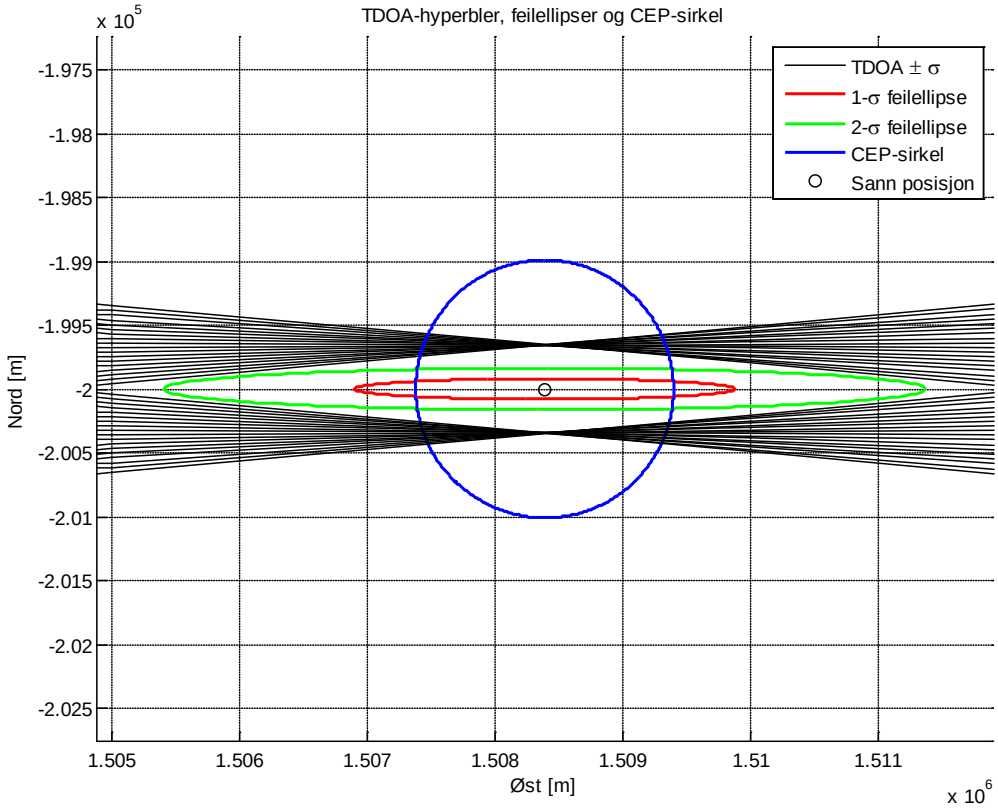
Estimeringsnøyaktigheten til radarene er avhengig av både avstanden mellom radaren og satellittbanen, og hvor i det opplyste området radaren ligger. Avstanden mellom radaren og satellittbanen påvirker spredningen til hyperblene beregnet av TDOA-ene, mens plasseringen av radaren i det opplyste området påvirker antallet mottatte hovedlobepasseringer. Figur 7.16 viser et tilfelle hvor sann TDOA er lik 0 ns, og hvor det er tegnet inn hyperbler for sann TDOA med måleavvik (pluss/minus et standardavvik). De blå hyperblene viser sann TDOA pluss/minus et standardavvik (σ = 70,71 ns), de røde hyperblene viser sann TDOA pluss/minus et standardavvik hvor 10 TDOA midles (σ = 22,36 ns), og de grønne hyperblene viser sann TDOA pluss/minus et standardavvik hvor 20 TDOA midles (σ = 15,81 ns). I figuren er satellittene plassert symmetrisk om origo langs y-aksen. Som figuren viser øker spredningen av hyperbler med målingsavvik når

avstanden mellom satellittene og radarene øker. Samtidig viser figuren også at ved å midle flere TDOA-er, reduseres spredningen til hyperblene.



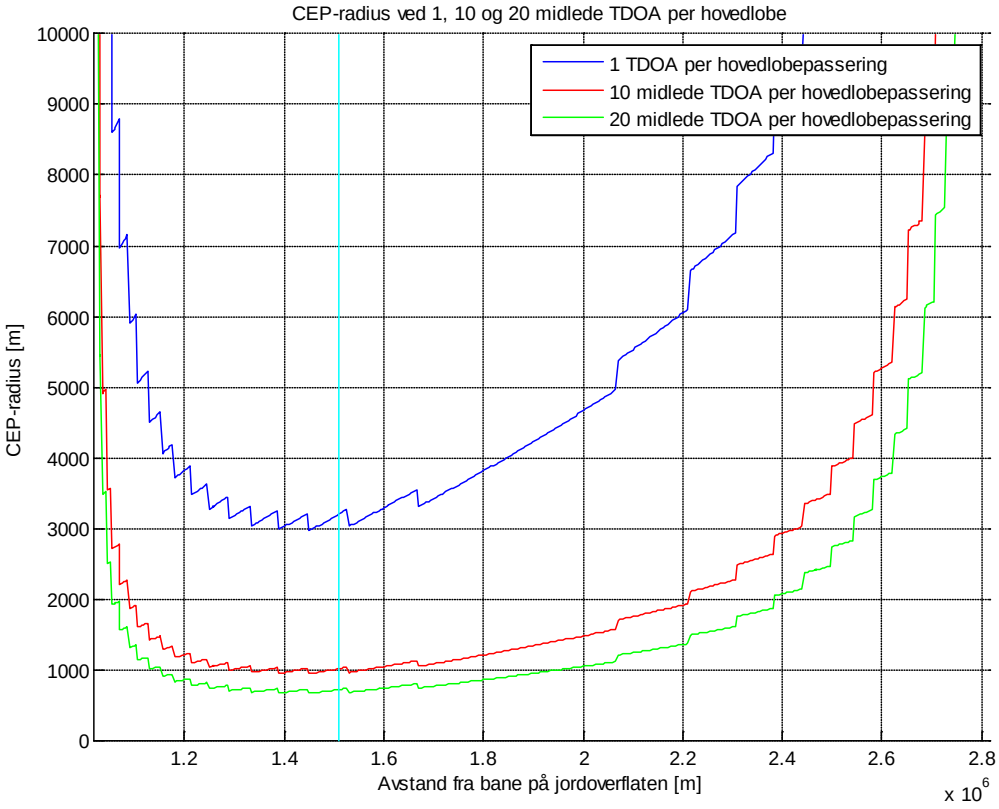
Figur 7.16: Sann hyperbel og hyperbler for sann TDOA pluss/minus et standardavvik. For blå hyperbler gjelder en TDOA per hovedlobepassering, røde hyperbler gjelder 10 midlede TDOA-er per hovedlobepassering og for grønne hyperbler gjelder 20 midlede TDOA-er per hovedlobepassering.

Som for avsnitt 7.1 antas det i CRLB at radarene har en rotasjonstid på 2,2 s, og at hver hovedlobepassering resulterer i 10 pulser som det beregnes TDOA mellom (altså 10 TDOA som midles per hovedlobepassering). Figur 7.17 viser 1σ feilellipse, 2σ feilellipse og CEP-sirkel for en radar som ligger midt i hovedlobene til satellittene, når det antas at satellitten mottar 18 hovedlobepasseringer fra radaren (fra Figur 7.15). I figuren er det også tegnet opp hyperbler for sann TDOA med målingsavvik (pluss/minus et standardavvik, hvor σ = 22,36 ns). Som figuren viser, er usikkerheten i posisjonen til radaren hovedsakelig i avstand, mens usikkerheten er liten i bredde (retning nord).



Figur 7.17: 1σ og 2σ feilellipser og CEP-sirkel når satellittene mottar 18 hovedlobepasseringer, hver det midles 10 TDOA, fra en radar midt i hovedloben til satellittene.

Estimeringsnøyaktigheten påvirkes av både spredningen til hyperblene, antall hovedlobepasseringer og antall midlede TDOA per hovedlobepassering. Figur 7.18 viser CEP-radius for radarer i ulike posisjoner i det opplyste området, for 1, 10 og 20 midlede TDOA per hovedlobepassering. Av figuren ses det at usikkerheten er høy nær endepunktene av det opplyste området. Dette skylles få mottatte hovedlobepasseringer fra radarer i disse områdene. Nøyaktigheten blir derfor størst i senter av det opplyste området, ettersom satellittene mottar flest hovedlobepasseringer fra radarer i nettopp disse områdene. Hoppene i CEP-radiusen kommer av at det legges til eller fjernes en hovedlobepassering, ettersom nøyaktigheten blir bedre når satellittene mottar en ekstra hovedlobepassering og nøyaktigheten blir dårligere når satellittene mottar færre hovedlobepasseringer. Stigningen til CEP-radiusen mellom hoppene skylles at usikkerheten øker når avstanden mellom satellittene og radarene øker. I figuren ses det at usikkerheten faller drastisk med å midle 10 TDOA per hovedlobepassering, kontra å bare anta en TDOA per hovedlobepassering. Samtidig er det kun en liten forbedring av å midle 20 TDOA per hovedlobepassering kontra 10 TDOA (standardavviket for 20 midlede TDOA er 71 % av standardavviket til 10 midlede TDOA – faktor på $1/\sqrt{2}$). Det betyr at CRLB ikke gir altfor dårlig nedre grense for estimeringsnøyaktigheten dersom det antas 10 midlede TDOA, dersom det egentlig er 20 midlede TDOA per hovedlobepassering.



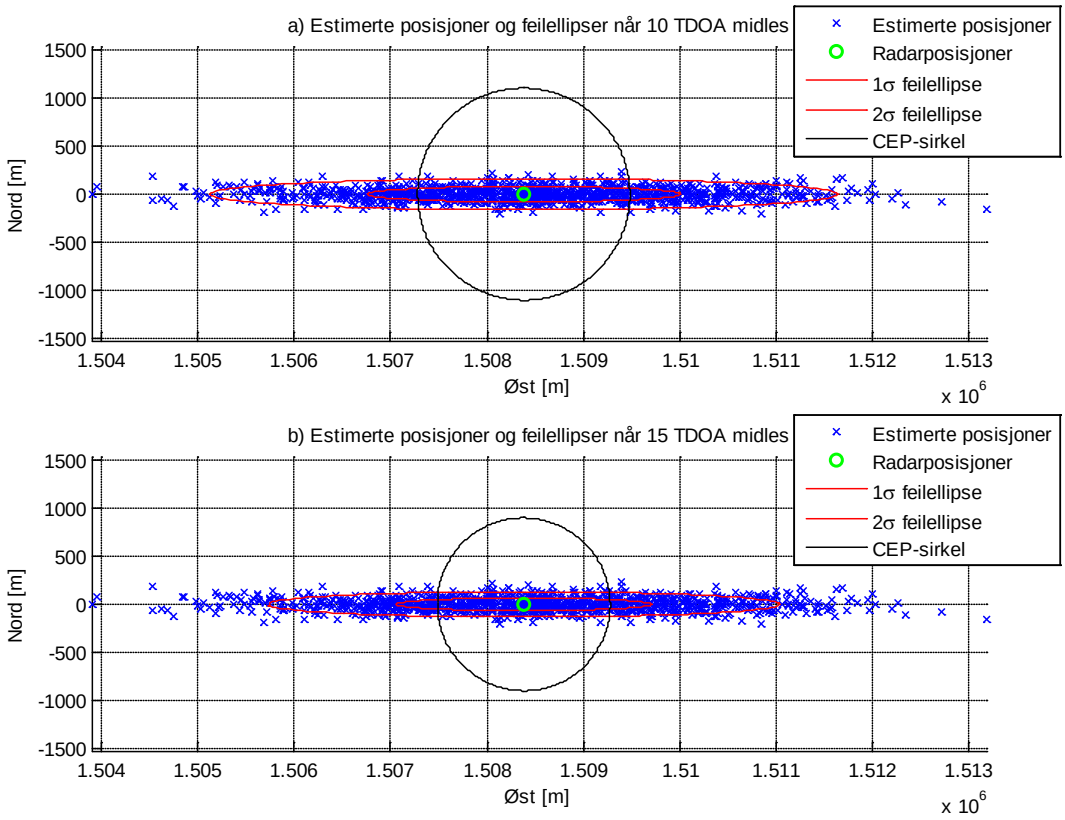
Figur 7.18: CEP-radius ved ulike avstander fra satellittbanen. Den sorte linjen er senter av hovedlobene til satellittene. Blå kurve viser en TDOA per hovedlobepassering, rød 10 midlede TDOA-er per hovedlobepassering og grønn 20 midlede TDOA-er per hovedlobepassering. Vertikal lys blå linje viser senter av satellittenes hovedlober.

MATLAB-koden til å beregne CRLB, feilellipser og CEP-sirkel for radarposisjonene i satellitt-scenarioet ligger i appendiks I.3.

7.2.2 Radarformasjon 1 – En radar

Som for UAS-scenarioet undersøkes det hvor godt algoritmen estimerer radarposisjoner når det ikke er noen tvil om hvilke pulser som hører sammen i de to sensorene, og hvilke pulser som kommer fra samme radar. Figur 7.19 a) viser estimerte posisjoner til radaren ved 1000 Monte Carlo gjennomkjøringer. I simuleringen er det simulert data fra en hel overflyvning. I figuren er det tegnet opp 1σ og 2σ feilellipser og en CEP-sirkel beregnet ved CRLB. I CRLB antas det her, som for UAS-scenarioet, at satellittene mottar 10 pulser det beregnes TDOA mellom. I simuleringen mottas det i snitt 15 pulser som det beregnes TDOA mellom og som midles. Dette fører til at CRLB gir høyere standardavvik enn hva spredningen til de estimerte posisjonene er i simuleringen. I figuren ligger 44,5 % av estimatene innenfor 1σ feilellipsen, 91,1 % innenfor 2σ feilellipsen og 50,5 % innenfor CEP-sirkelen. Som for UAS-scenarioet er andelen estimat innenfor de ulike feilellipsene høyere enn hva CRLB tilsier. Ved å øke antall TDOA per midlet TDOA til 15, burde andelen estimat som ligger innenfor de ulike feilellipsene stemme bedre overens med CRLB. Figur 7.19 b) viser samme simulering som i a), men hvor 15 TDOA midles. I figuren ligger 30,7 % av estimatene innenfor 1σ ellipsen, 79,2 % innenfor 2σ ellipsen og 42,2 % innenfor CEP-sirkelen. Andelen estimat innenfor ellipsene ligger lavere enn hva som er teoretisk oppnåelig, og som for UAS-scenarioet skylles dette at estimeringsalgoritmen velger ut TDOA

som ligger innenfor pluss/minus to standardavvik fra de sanne TDOA-ene til den initielle posisjonen.

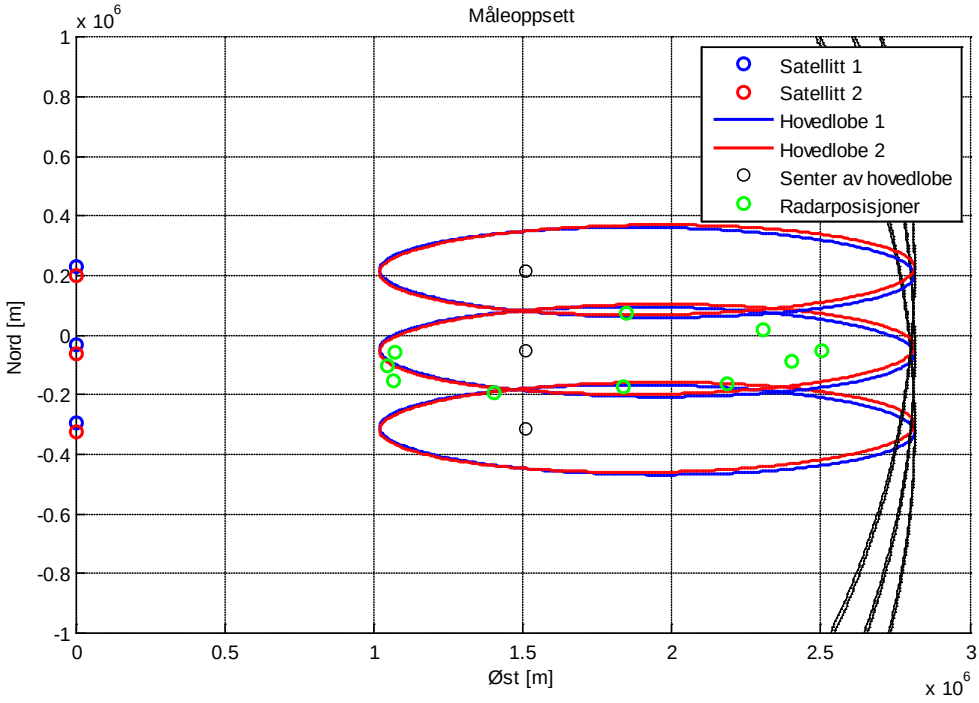


Figur 7.19: Estimerte posisjoner av en radar i satellitt-scenarioet fra 1000 gjennomkjøringer av Monte Carlo simulering, 1σ og 2σ feilellipser og CEP-sirkel. I a) er 10 TDOA midlet per hovedlobepassering, og i b) 15 TDOA midlet per hovedlobepassering.

Simuleringen viser samme resultat som i avsnitt 7.1.2 hvor det ble analysert estimeringsnøyaktigheten for en radar i UAS-scenarioet. Monte Carlo simuleringen viser at spredningen er større enn hva som er teoretisk oppnåelig, men at estimatoren er forventningsrett.

7.2.3 Radarformasjon 2 – Ti radarer spred utover hele hovedloben til satellittene

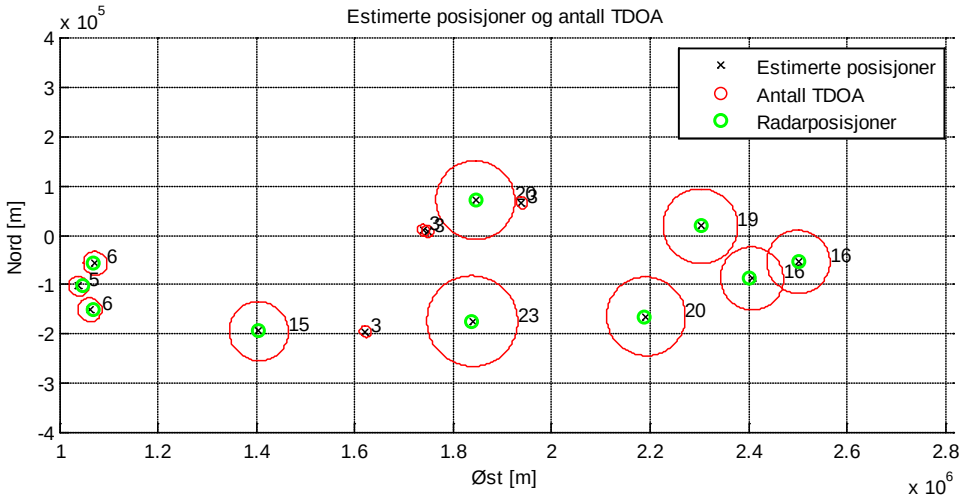
I radarformasjon 2 er 10 radarer plassert utover hele det opplyste området. Figur 7.20 viser radarenes posisjoner, samt satellittenes posisjon og deres hovedlober i tidspunktet når den første radaren kommer inn i det opplyste området, i tidspunktet når den siste radaren går ut av det opplyste området, og i tidspunktet midt mellom første og siste tidspunkt. Mot øst i figuren er det plottet horisonten til den kuleformede jorden. Ved oppsettet nedenfor tar det totalt 78 s fra første radar kommer inn i det opplyste området til den siste radaren er ute av det opplyste området.



Figur 7.20: Satellitt-posisjoner og deres hovedlober nede på jordoverflaten ved tre tidspunkter, og ti radarer.

En simulering

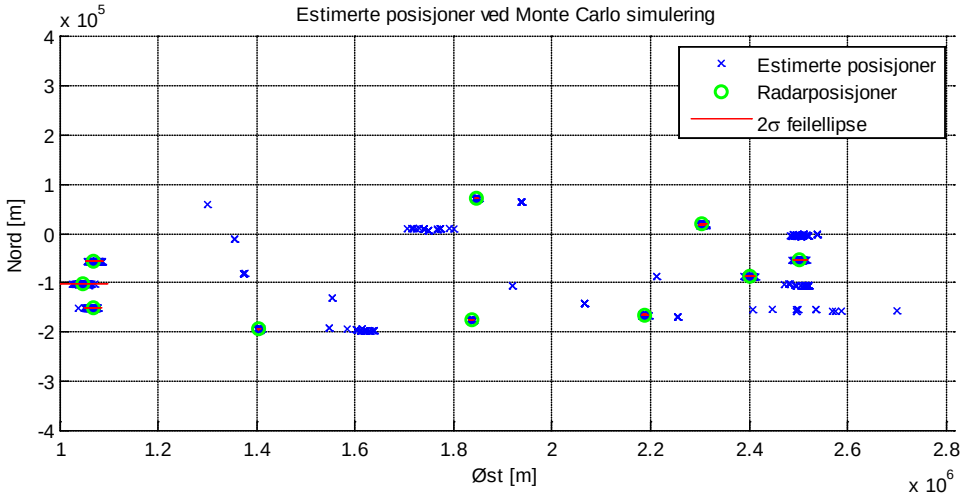
Analyserer først formasjonen ved å kjøre algoritmen en gang. Figur 7.21 viser 14 estimerte posisjoner når sensorene mottar pulser fra 10 radarer, hvorav alle radareneposisjonene blir estimert. I figuren er det også tegnet opp antall TDOA som er benyttet i hvert estimat. Som figuren viser, varierer antall TDOA etter hvor i det opplyste området radarene ligger. Dvs. at estimatene av radarene som ligger helt til venstre i figuren har færre TDOA enn estimatene i midten av figuren. Dette stemmer også med Figur 7.15, som viser at forventet antall hovedlobe-passeringer (ideelt sett antall midlede TDOA) er lavere for radarer nært endepunktene av det opplyste området enn for radarer i senter av det opplyste området.



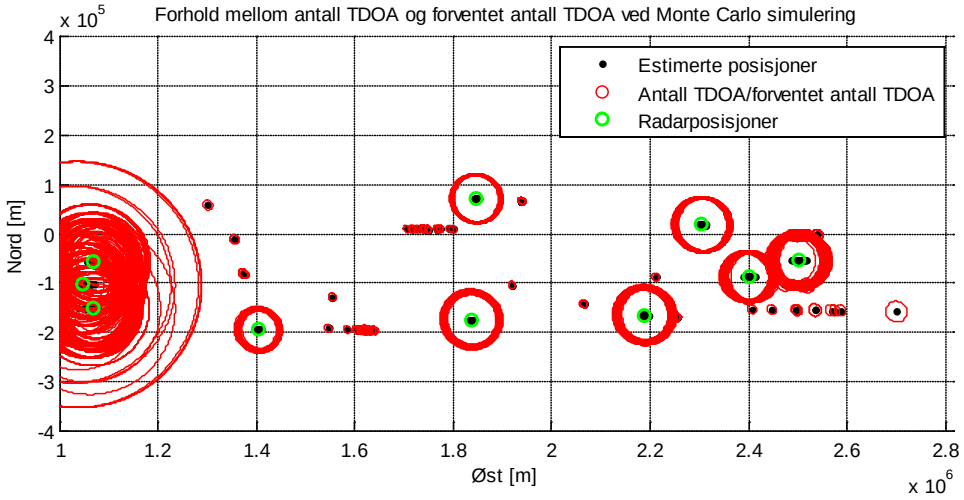
Figur 7.21: Estimerte posisjoner fra en simulering, ti radarposisjoner og antall TDOA benyttet i hvert estimat (radius lik antall TDOA multiplisert med 4 km).

Monte Carlo simulering

Kjører Monte Carlo simulering med 100 gjennomkjøringer. Figur 7.22 viser estimerte posisjoner for de 100 gjennomkjøringene. I simuleringen gir hver gjennomkjøring estimater som tilhører hver av de 10 radarene, i tillegg til noen gale estimat. Ettersom radarene er spredt utover hele det opplyste området, vil antall hovedlobepasseringer fra de ulike radarene variere. Det gir derfor lite mening å se på antall TDOA i hvert estimat for å skille riktige og gale estimat. Figur 7.23 viser forholdet mellom antall midlede TDOA per estimat og forventet antall midlede TDOA fra Figur 7.15. Som figuren viser er forholdet mye høyere for estimatene nært radarposisjonene enn for de gale estimatene, og det burde være forholdsvis greit å sortere de riktige estimatene fra de gale.



Figur 7.22: Estimerte posisjoner fra 100 gjennomkjøringer av Monte Carlo simulering, 2σ feilellipser og ti radarposisjoner.

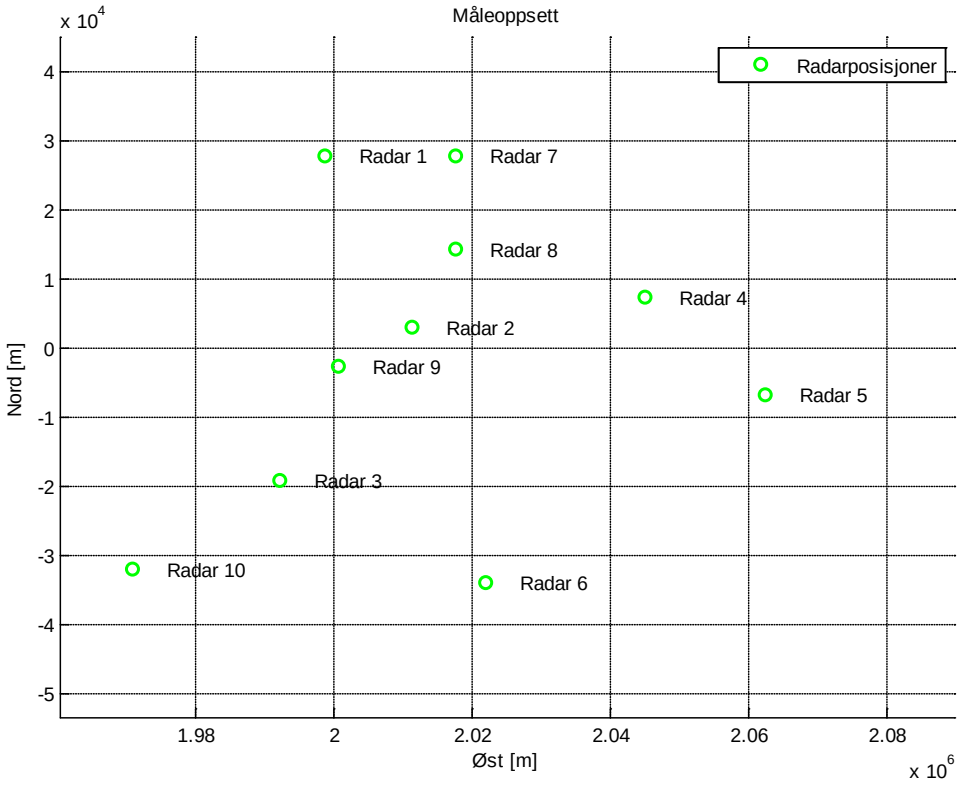


Figur 7.23: Forhold mellom antall TDOA benyttet i hvert estimat og forventet antall TDOA fra 100 gjennomkjøringer av Monte Carlo simulering (radius lik forholdet multiplisert med 50 km).

Simuleringen har vist at algoritmen gir estimat av radarposisjoner som skiller seg fra gale estimat når ti radarer ligger spredd utover hele det opplyste området.

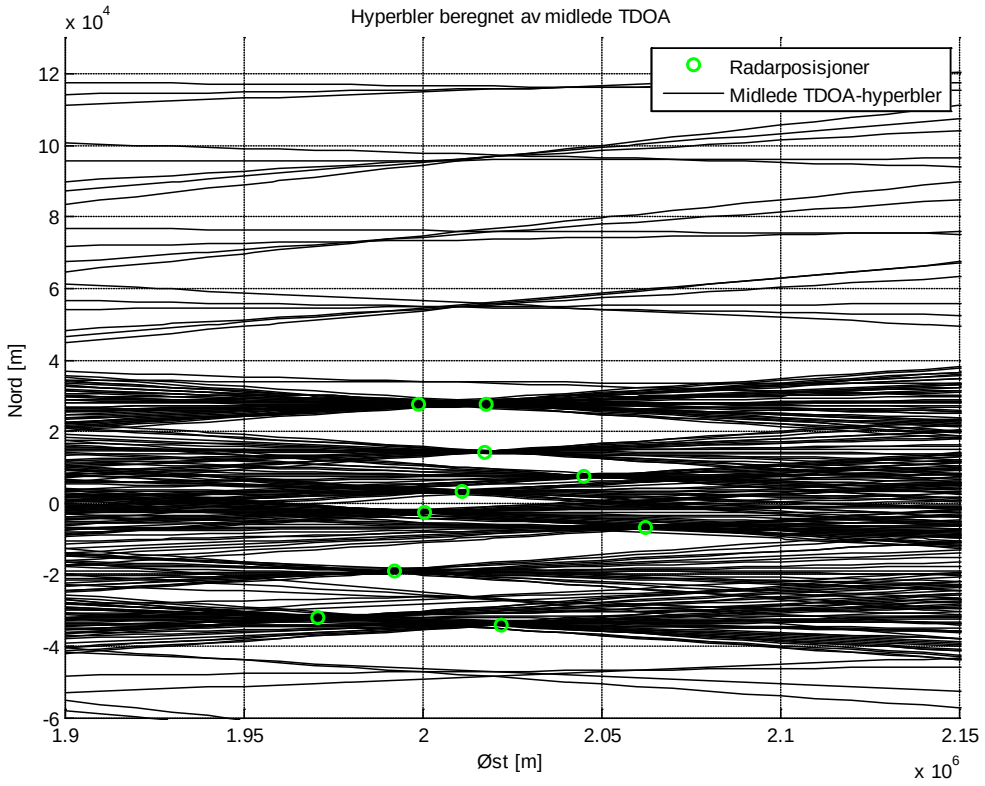
7.2.4 Radarformasjon 3 – Ti radarer plassert tett midt i hovedloben til satellittene

Plasserer ti navigasjonsradarer innenfor et mindre område midt i det opplyste området. Ettersom alle radarene ligger i omtrent samme avstand fra satellittene, forventes det at satellittene motter 20 hovedlober fra alle radarene. Figur 7.24 viser posisjonen til radarene.



Figur 7.24: Ti radarer plassert tett i senter av opplyste området til satellittene.

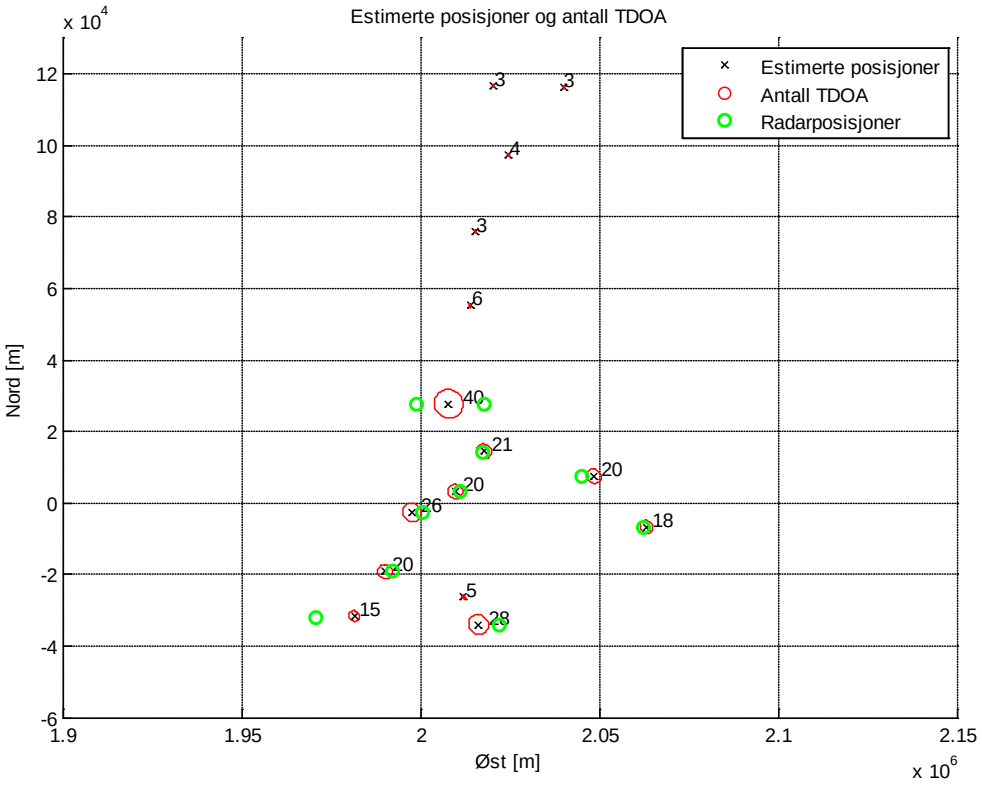
Figur 7.25 viser hyperbler beregnet fra midlede TDOA i løpet av en overflyvning med satellittene fra en gjennomkjøring av Monte Carlo simuleringen. I figuren er det et par potensielle problemer for estimeringsalgoritmen. Radar 1 og radar 7 ligger på en linje slik at mange av hyperblene som tilhører radar 1 går gjennom eller nært radar 7, og vice versa. Dette kan føre til at algoritmen får problemer med å estimere disse to posisjonene. Videre er det mange hyperbler som skjærer hverandre der det ikke er noen radarer. Dette kan danne gale estimater. Forhåpentligvis er antallet TDOA-er for disse estimatene mye mindre enn for de riktige estimatene.



Figur 7.25: Hyperbler beregnet fra midlede TDOA fra ti radarer i senter av opplyste området til satellittene.

En simulering

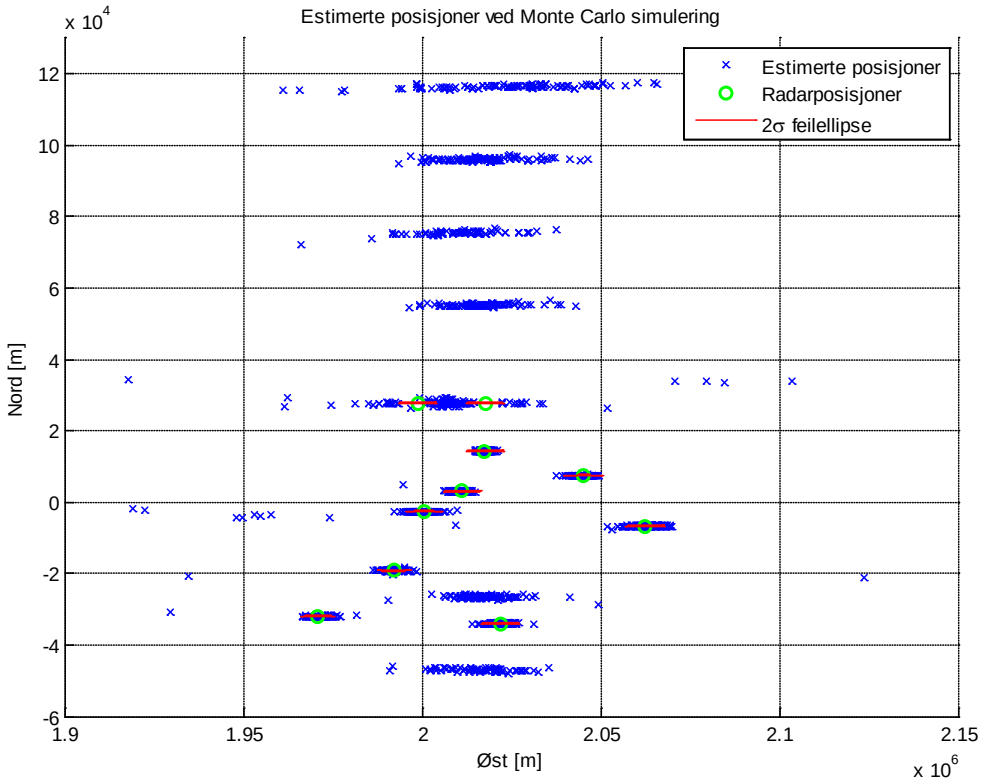
Figur 7.26 viser 20 estimerte posisjoner når det mottas pulser fra 10 radarer. Figuren viser også antall midlede TDOA som er benyttet i hvert estimat. I simuleringen tilhører 8 av estimatene tydelig hver sine radarer, mens radar 1 og 7 får et felles estimat midt mellom radarene. Dette skylles at radarene ligger så tett at mange av TDOA-ene til den ene radaren også passer godt med den andre radaren, og algoritmen klarer dermed ikke å skille disse to radarene. Geometrisk betyr dette at hyperblene fra radar 1 går gjennom eller nært radar 7, og vice versa, noe som ses tydelig i Figur 7.25. Allikevel ses det av Figur 7.26 at dette estimatet har 40 midlede TDOA, noe som er dobbelt så høyt som det forventede. Dette kan skylles at det er en radar som roterer raskt og gir mange hovedlobepasseringer, at estimatet stjeler mange TDOA fra andre radarer, eller nettopp at estimatet tilhører to radarer. Videre ligger det en del gale estimater nord for radar 1 og 7. Fra Figur 7.25 ses det at disse estimatene kommer fra hyperbler som ikke tilhører noen radar.



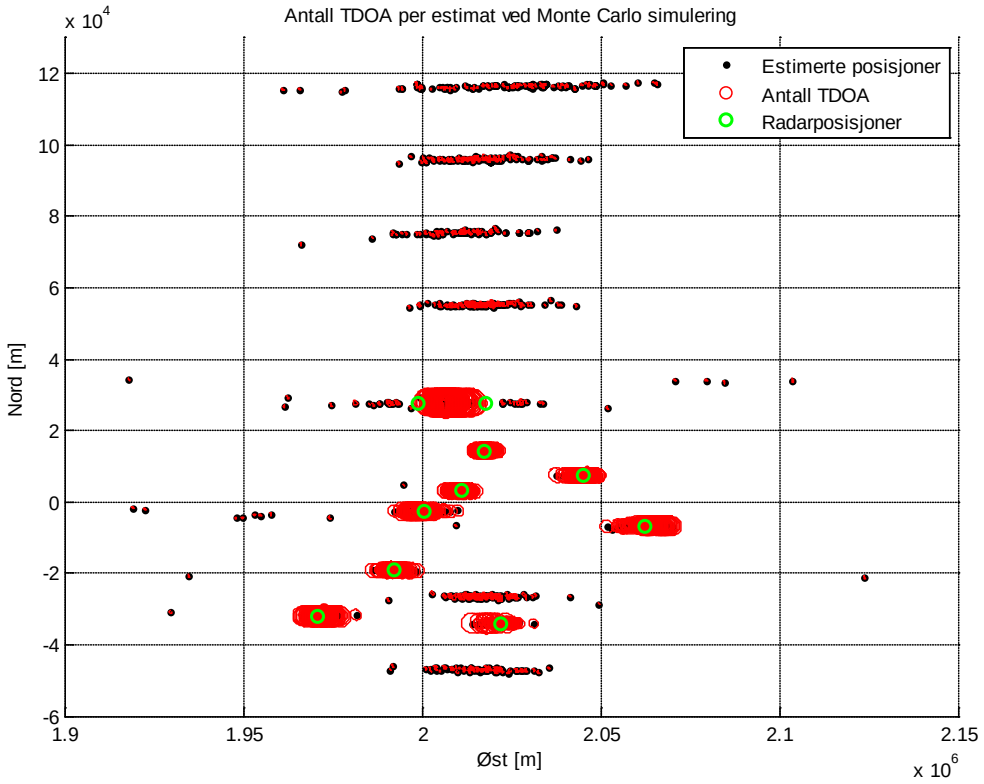
Figur 7.26: Estimerte posisjoner fra en simulering, ti radarposisjoner og antall TDOA benyttet i hvert estimat (radius lik antall TDOA multiplisert med 100 m).

Monte Carlo simulering

Figur 7.27 viser estimerte posisjoner ved 100 gjennomkjøringer av en Monte Carlo simulering. Algoritmen gir for hver gjennomkjøring et estimat for 8 av radarene, og stort sett ett felles estimat for radar 1 og 7. I tillegg ses det noen linjer med gjentakende gale estimat som skylles at mange hyperbler ligger i disse områdene, se Figur 7.25. Figur 7.28 viser antall midlede TDOA som er benyttet i estimeringen av posisjonene i Monte Carlo simuleringen. Det ses tydelig i figuren at antall TDOA for estimatene nært radarene er langt høyere enn for de gale estimatene, og det burde derfor være greit å skille de riktige estimatene ut fra de gale estimatene.

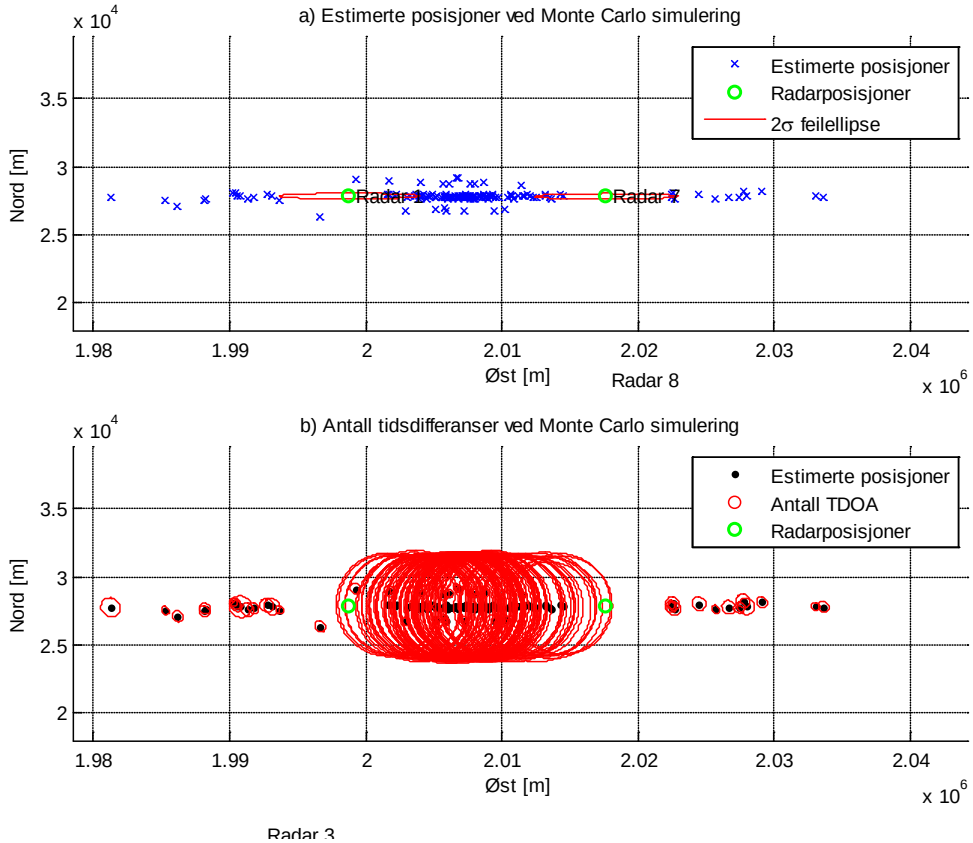


Figur 7.27: Estimerte posisjoner fra 100 gjennomkjøringer av Monte Carlo simulering, 2σ feilellipser og seks radarposisjoner.



Figur 7.28: Antall TDOA benyttet i hvert estimat fra 100 gjennomkjøringer av Monte Carlo simulering (radius lik antall TDOA multiplisert med 100 m).

Figur 7.29 viser et utsnitt av radar 1 og 7 i Figur 7.27 og Figur 7.28. Figur a) viser de estimerte posisjonen og b) viser antall TDOA per estimat. Som figuren viser havner nesten alle estimatene mellom radarene. I simuleringen ligger estimatene av radar 1 og 7 mellom radarene, og det ser ut til at radarene ligger for tett til at algoritmen skal gi riktige estimater av radarene.

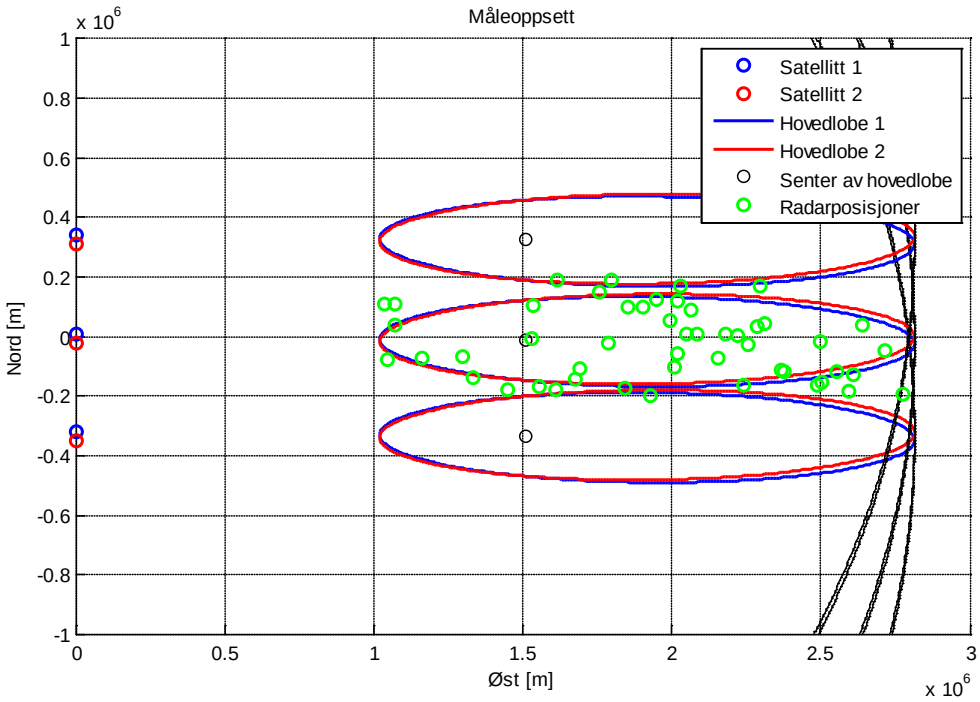


Figur 7.29: a) Estimerte posisjoner fra 100 gjennomkjøringer av Monte Carlo simulering, 2σ feilellipser og seks radarposisjoner. b) Antall TDOA benyttet i hvert estimat fra 100 gjennomkjøringer av Monte Carlo simulering (radius lik antall TDOA multiplisert med 100 m).

Simuleringen av ti radarer tett plassert midt i det opplyste området til satellittene viser at algoritmen gir riktige estimat av radarer som ligger forholdsvis tett i bredde (i retning nord), men har problemer med å estimere riktige estimat for radarer som ligger tett i avstand (i retning øst). Ved å sammenlikne med simuleringen fra avsnitt 7.2.3 er det flere gale estimat når radarene ligger tett. Hvor tett to radarer kan ligge for at algoritmen skal gi estimater av begge radarer analyseres nærmere i kapittel 8.

7.2.5 Radarformasjon 4 – 50 radarer spred utover hele hovedloben til satellittene

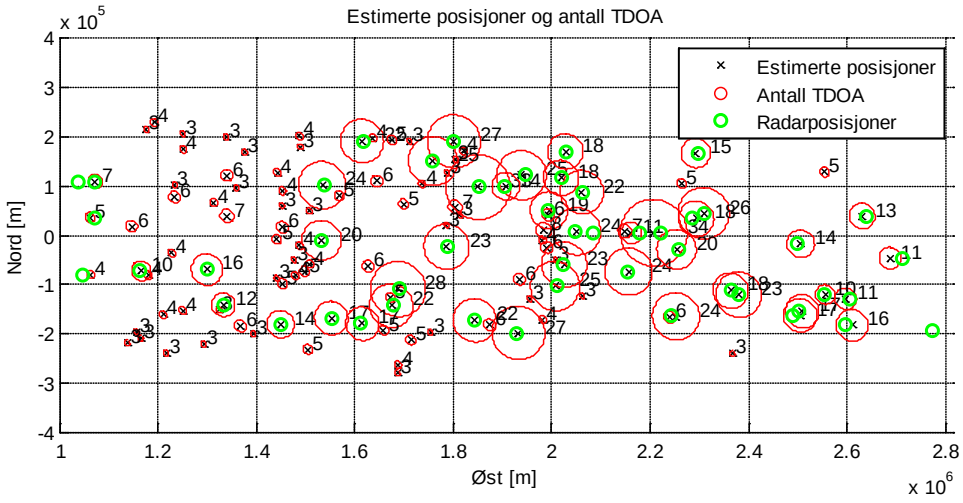
I radarformasjon 4 undersøkes det hvordan algoritmen takler å geolokalisere 50 radarer. Dette fører til at satellittene mottar pulser fra mange radarer samtidig. Algoritmen får da sannsynligvis problemer med å assosiere pulsene i de to sensorene, noe som fører til mange gale TDOA-er. I tillegg vil algoritmen også sannsynligvis få problemer med å assosiere TDOA-ene fra samme radarer, noe som fører til mange gale estimat. Figur 7.30 viser radarenes posisjoner, samt satellittenes posisjon og deres hovedlober i tidspunktet når den første radaren kommer inn i det opplyste området, i tidspunktet når den siste radaren går ut av det opplyste området, og i tidspunktet midt mellom første og siste tidspunkt. Mot øst i figuren er det plottet horisonten til den kuleformede jorden.



Figur 7.30: Satellitt-posisjoner og deres hovedlober nede på jordoverflaten ved tre tidspunkter, og 50 navigasjonsradarer.

En simulering

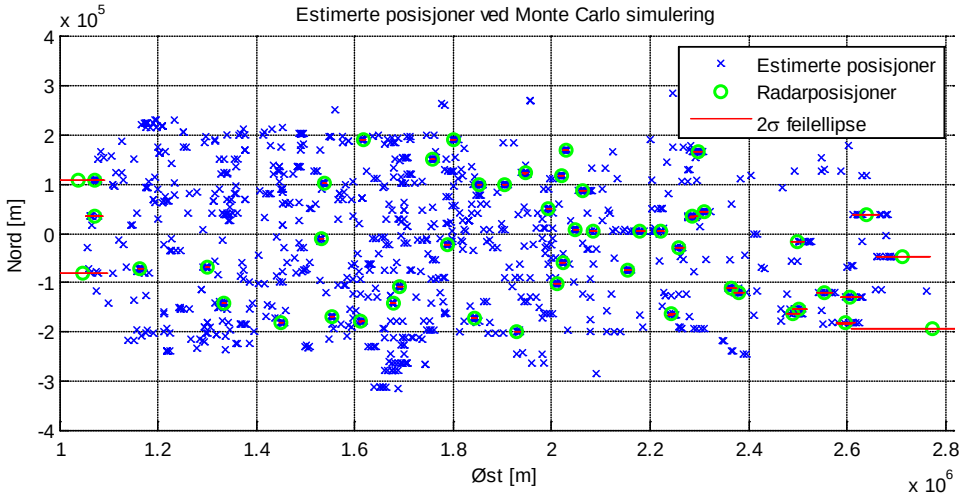
Analyserer først formasjonen ved å kjøre algoritmen en gang. Figur 7.31 viser 123 estimerte posisjoner når sensorene mottar pulser fra 50 radarer, hvorav 43 av estimatene tydelig tilhører hver sin radar. De resterende estimatene er da enten gale estimater eller estimater som tilhører to eller flere radarer. I figuren er det også tegnet inn røde sirkler for antall TDOA som er benyttet i hvert estimat.



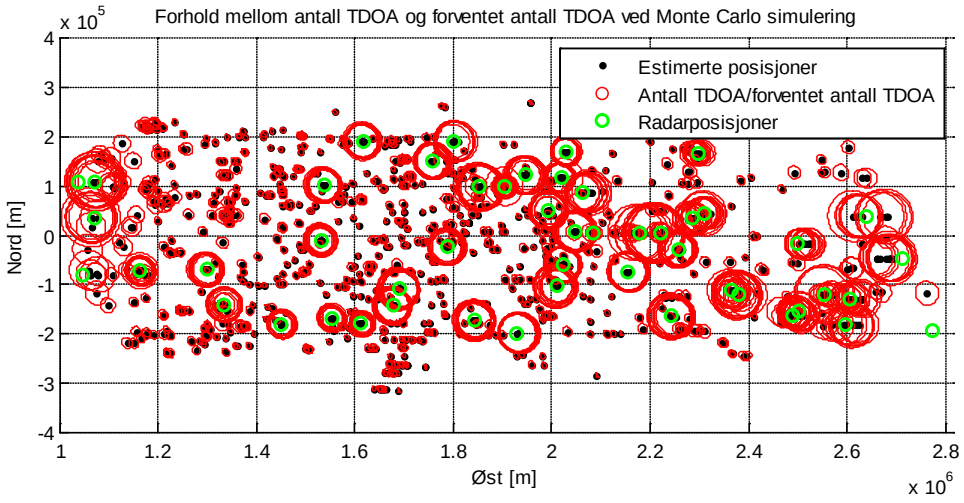
Figur 7.31: Estimerte posisjoner fra en simulering, 50 radarposisjoner og antall TDOA benyttet i hvert estimat (radius lik antall TDOA multiplisert med 2 km).

Monte Carlo simulering

Undersøker radarformasjonen med en Monte Carlo simulering med 10 gjennomkjøringer,. Det benyttes kun 10 gjennomkjøringer ettersom beregningstiden er svært lang. Dette diskuteres nærmere i avsnitt 10.4. Figur 7.32 viser estimerte posisjoner for de 10 gjennomkjøringene. Fra figuren ser det ut til at de estimerte posisjonene havner over hele området. Figur 7.33 viser forholdet mellom antall TDOA per estimat og forventet antall mottatte TDOA. I figuren skiller mange av de gale estimatene seg ut ved at det benyttes få TDOA i estimatet i forhold til hva som forventes å motta fra en radar som ligger i den estimerte posisjonen.

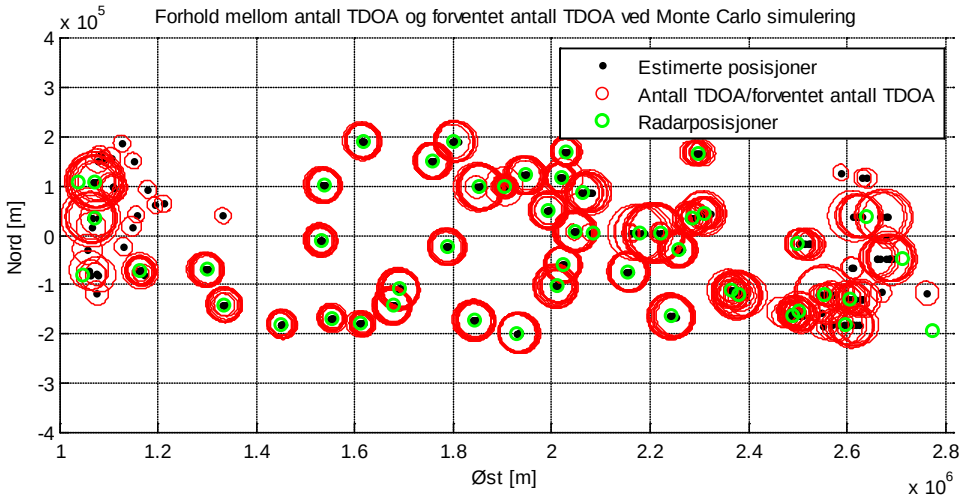


Figur 7.32: Estimerte posisjoner fra 10 gjennomkjøringer av Monte Carlo simulering, 2σ feilellipser og 50 radarposisjoner.



Figur 7.33: Forhold mellom antall TDOA benyttet i hvert estimat og forventet antall TDOA fra 10 gjennomkjøringer av Monte Carlo simulering (radius lik forholdet multiplisert med 30 km).

I Figur 7.33 er det vanskelig å skille riktige estimat fra de gale estimatene på grunn av høyt antall estimater. Figur 7.34 viser forholdet mellom antall TDOA per estimat og forventet antall mottatte TDOA, men hvor forholdet må være større enn 0,5. Dvs. at figuren viser estimat hvor antallet TDOA minst er halvparten av hva som forventes å motta, og antall gale estimat reduseres betraktelig. Som figuren viser ligger nesten alle estimatene i nærheten av radarposisjonene. I gjennomsnitt estimeres det 49 estimat per gjennomkjøring, hvorav 41 av radarposisjonene blir estimert i hver gjennomkjøring med et estimat som tilhører den posisjonen. I figuren ser det også ut til at de fleste gale estimatene havner nært endepunktene av det opplyste området. Disse gale estimatene har ikke flere TDOA enn gale estimat ellers i det opplyste området, men siden det forventes færre mottatte hovedlobepasseringer fra radarer nært endepunktene blir forholdet mellom antall TDOA og forventet TDOA høyere for disse estimatene. Allikevel kan det være vanskelig å skille de riktige estimatene nært endepunktene og de gale estimatene, ettersom de riktige estimatene består av få TDOA.



Figur 7.34: Forhold mellom antall TDOA og forventet antall som er større enn 0,5 fra 10 gjennomkjøringer av Monte Carlo simulering (radius lik forholdet multiplisert med 30 km).

Simuleringen av 50 radarer plassert utover hele det opplyste området gir et komplisert signalmiljø, hvor begge satellittene mottar pulser fra mange radarer samtidig. I simuleringen gir gjennomkjøringene i snitt 123 estimater, hvorav rundt 41 av estimatene tilhører hver sin radar. Ettersom antallet hovedlobepasseringer fra hver radar varierer etter hvor i hovedloben radarene ligger, kan ikke antall TDOA benyttes direkte til å bestemme om estimatet er riktig eller galt. Derimot kan forholdet mellom antall TDOA i estimatet og forventet antall TDOA gi en pekepinn på om estimatet er riktig. I Figur 7.33 ble det satt at dette forholdet måtte være større enn 0,5. Dette førte til at mange av de gale estimatene ble luket bort og det ble i gjennomsnitt estimert 49 estimat for hver gjennomkjøring.

8 Analyse av hvor tett to radarer kan ligge for at algoritmen skal estimere begge radarposisjonene

Analysene i kapittel 7 viste at noen typiske radarformasjoner hvor algoritmen gav et estimat for to eller flere radarer. I dette kapittelet analyseres det hvor tett to radarer kan ligge og at algoritmen allikevel skal gi estimater som tydelig tilhører hver av radarene. I analysene undersøkes det kun to radarer, hvor de enten ligger på en linje som står normalt på sensorbanen eller at de ligger på en linje som er parallell med sensorbanen. I analysen benyttes det Monte Carlo simuleringer med 100 gjennomkjøringer.

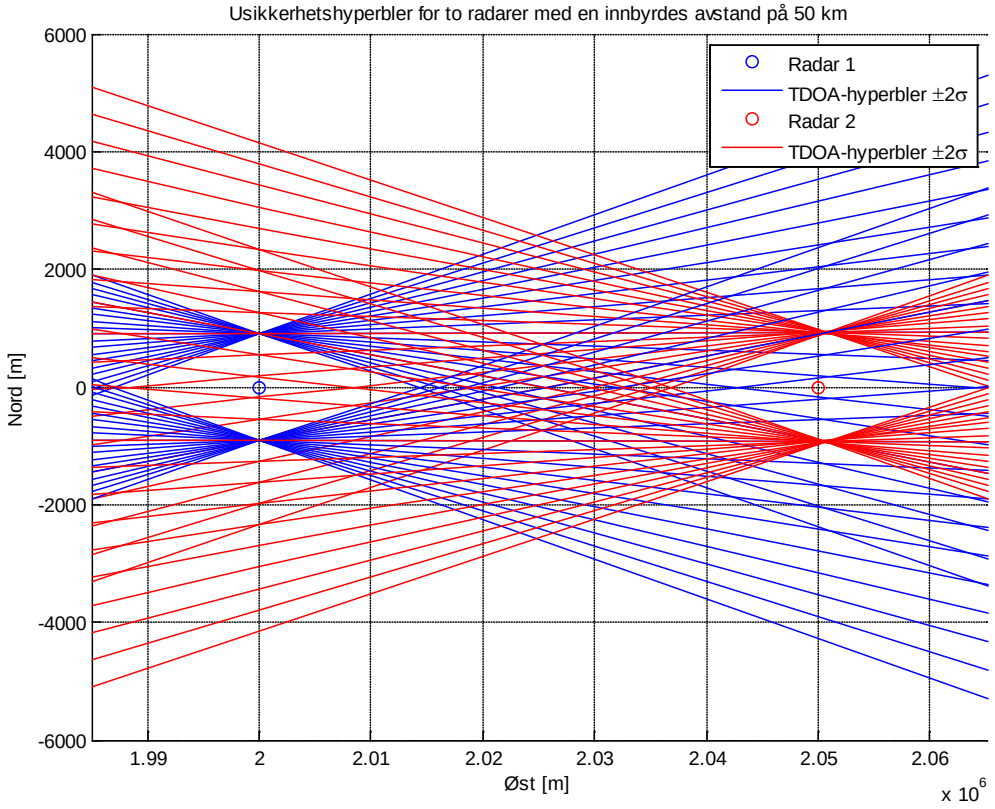
Analysen i kapittelet er delt inn i to deler; i avsnitt 8.1 ses på hvor tett to radarer to radarer kan ligge i satellitt-scenarioet og i avsnitt 8.2 ses det på hvor tett to radarer kan ligge i UAS-scenarioet. Grunnen til at satellitt-scenarioet analyseres først er fordi satellittene mottar målinger i løpet av en hel overflyvning. Minimumsavstanden mellom radarene påvirkes da bare av avstanden mellom satellittbanen og radarene, og det er kanskje enklere å bestemme hvor tett radarene kan ligge. For UAS-scenarioet vil minimumsavstanden mellom radarene i tillegg påvirkes av hvor radarenes posisjon i forhold til UAS-ene.

8.1 Geolokalisering av radarer i scenario med to satellitter

Analysen av minimumsavstanden mellom to radarer i satellitt-scenarioet er delt inn i to deler; a 8.1.1 hvor radarene ligger på en linje som står normalt på satellittbanen, og avsnitt 8.1.2 hvor radarene ligger på en linje som er parallell med satellittbanen.

8.1.1 To radarer som ligger på en linje som står normalt på satellittbanen

Når radarene ligger på en linje som står normalt på satellittbanen, vil mange av TDOA-ene fra den ene radaren også kunne komme fra den andre radaren. Geometrisk betyr det at hyperblene mot den ene radaren gå gjennom eller nært den andre radaren. For å kunne skille radarene fra hverandre, må radarene ligger så langt fra hverandre at flest mulig av TDOA til den ene radaren ikke kan tilhøre den andre radaren. Figur 8.1 viser to radarer som ligger på linjen i x = 0 m. Benytter samme antakelser om radarene som i avsnitt 7.2. Dvs. at det forventes at satellittene motter 19 hovedlobepasseringer fra hver av radarene, hver med 10 pulser som det beregnes TDOA mellom. I figuren er det tegnet opp hyperbler for sann TDOA pluss/minus to standardavvik (for 10 midlede TDOA, σ = 22,36 ns). Grunnen til at det er tegnet inn for pluss/minus to standardavvik, er at estimatoren velger ut TDOA som ligger innenfor pluss/minus to standardavvik fra en initiell posisjon.I figuren kan satellittene motta 10 hovedlobepasseringer fra radar 2 (rød) hvor TDOA med et målingsavvik på to standardavvik ikke ligger innenfor TDOA med et målingsavvik på to standardavvik fra radar 1 (blå). Det betyr at dersom estimatoren starter med en initiell posisjon nært radar 1, så vil radar 2 fortsatt estimeres med TDOA-er fra 10 hovedlobepasseringer.



Figur 8.1: To radarer som ligger på linjen som står normalt på satellittbanen og tilhørende hyperbler beregnet av sann TDOA pluss/minus to standardavvik.

For at estimatoren skal skille to radarer fra hverandre, er det nødvendig at radarene ligger så langt fra hverandre at nok TDOA fra den ene radaren ikke tas med i estimatet av den andre radaren. Dvs. at flest mulig av TDOA-ene fra den ene radaren ligger utenfor pluss/minus to standardavvik fra sann TDOA til den andre radarens estimat. For å se hvor mange TDOA som satellittene må motta fra hver av radarene som ikke blandes med den andre radarens TDOA, simuleres to radarer ved ulike avstander mellom dem. Figur 8.2 viser 8 figurer, hvor hver rad viser ulike avstander mellom radarene. Kolonnen til venstre viser estimerte posisjoner fra Monte Carlo simuleringer og kolonnen til høyre viser antall TDOA benyttet i estimatene. Hver Monte Carlo simulering består av 100 gjennomkjøringer.

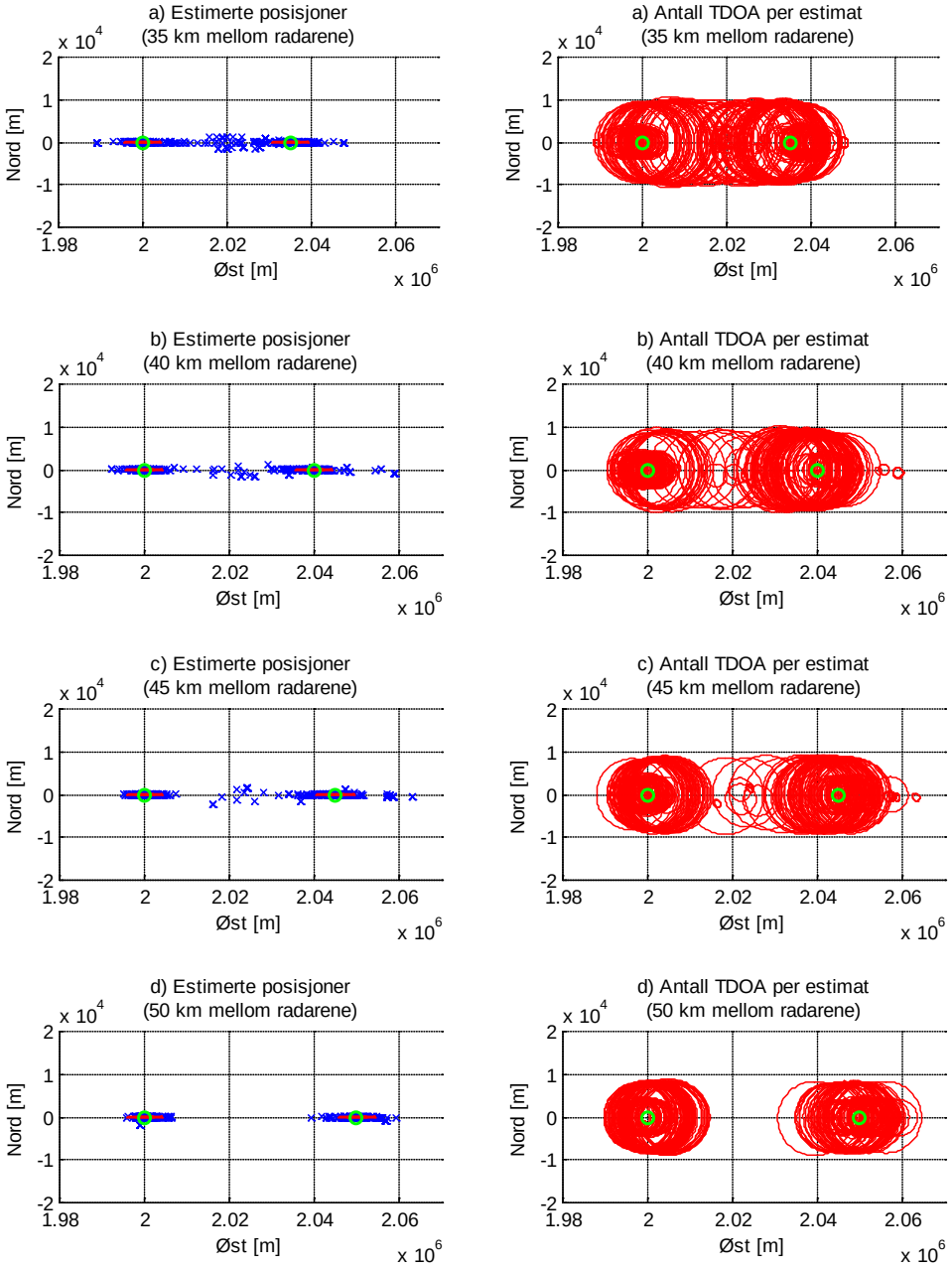
Radaroppsett a
I radaroppsett a) er to radarer plassert med en innbyrdes avstand på 35 km. I figuren ligger mange av estimatene mellom radarene og antallet TDOA som er benyttet i estimatene er like høyt for estimat nært radarene som midt i mellom radarene. Radarene ligger for tett til at algoritmen skal kunne gi to estimat som tydelig tilhører hver sin radar.

Radaroppsett b
I radaroppsett b) er to radarer plassert med en innbyrdes avstand på 40 km. Ved å sammenlikne med radaroppsett a), er antall estimat mellom radarene er redusert noe. Allikevel ligger radarene for tett til at algoritmen skal gi to estimat som tydelig tilhører hver sin radar.

Radaroppsett c
I radaroppsett c) er to radarer plassert med en innbyrdes avstand på 45 km. Fortsatt gir algoritmen noen gale estimater mellom radarene, men antallet er redusert betraktelig fra radaroppsett a) og b).

Radaroppsett d

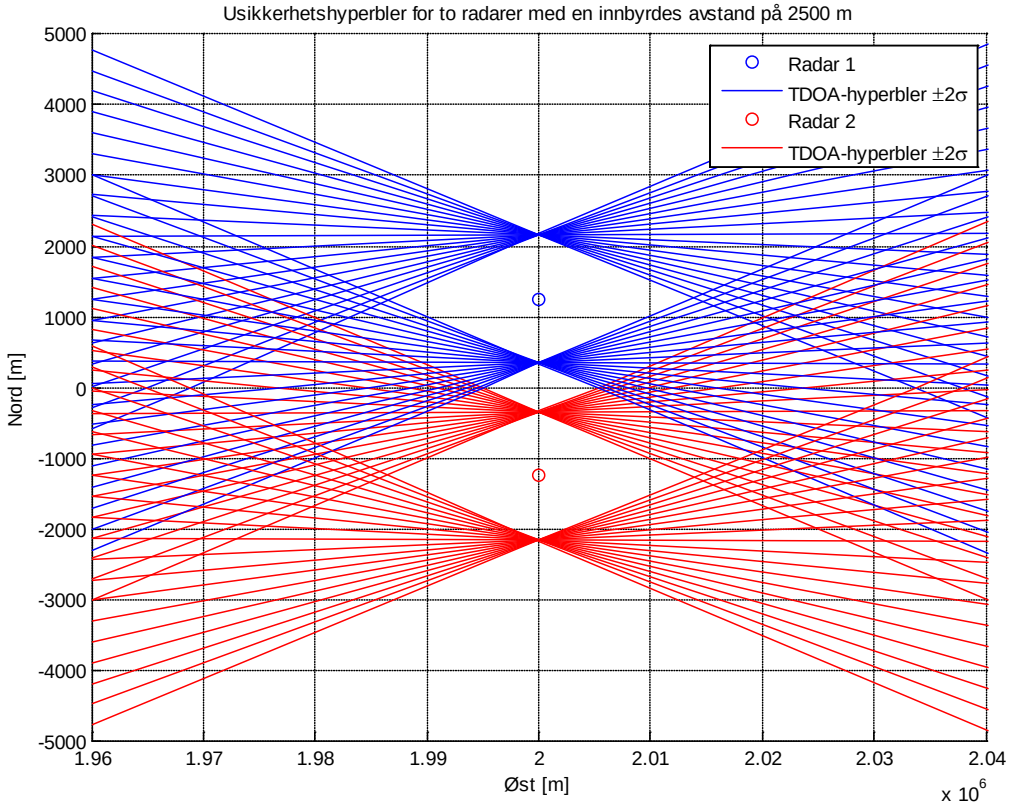
I radaroppsett d) er to radarer plassert med en innbyrdes avstand på 50 km. I figuren er det ingen estimat mellom radarene, og algoritmen gir to estimat som tydelig tilhører hver sin radar. Ved å se på antall TDOA som er benyttet i hvert estimat, har sirklene har to radier. Dette skylles at dersom radar 1 estimeres først, stjeler denne mange TDOA som tilhører radar 2. Og dersom radar 2 estimeres først, stjeler denne mange TDOA som tilhører radar 1. Allikevel er antall gjenværende TDIA, etter at første radar er estimert, høyt nok til å gi et godt estimat av den gjenværende radaren.



Figur 8.2: Fire tilfeller med to radarer som ligger på linjen som står normalt på satellittbanen. Kolonnen til venstre viser estimerte posisjoner (blå kryss) fra en Monte Carlo simulering, 2σ feilellipser (røde ellipser) og to radarposisjoner (grønne sirkler). Kolonnen til høyre viser antall TDOA benyttet i hvert estimat (radius lik antall TDOA multiplisert med 300 m).

8.1.2 To radarer som ligger på en linje som er parallell med satellittbanen

Når to radarer ligger på en linje som er parallell med satellittbanen, vil mange av TDOA-ene fra de to radarene danne falske mål med TDOA-er som kommer fra begge radarene. Geometrisk betyr det at hyperblene fra den ene radaren skjære hyperblene fra den andre radaren, og potensielt kan lage områder hvor mange hyperbler krysses og dermed gale estimater. For å forhindre at falske estimater opprettes, må radarene ligge så langt unna hverandre at færrest mulig TDOA-er fra begge radarene danner områder som mange TDOA-er kan komme fra. Figur 8.3 viser to radarer som ligger på linjen x = 2000 km. Det forventes å motta 19 hovedlobepasseringer fra begge radarene, hver med 10 pulser som det beregnes TDOA mellom. I figuren er det tegnet opp hyperbler for sann TDOA pluss/minus to standardavvik (for 10 midlede TDOA, σ = 22,36 ns). I figuren ligger radarene så langt fra hverandre at dersom estimatoren starter i en posisjon nært en av radarene vil den ikke ta med noen TDOA som tilhører den andre radaren. Derimot hvis estimatoren starter i en initiell posisjon i (0 km nord, 1980 km øst) eller (0 km nord, 2020 km øst) vil estimatoren kunne ta med 8 av hovedlobepasseringene til både radar 1 og radar 2.



Figur 8.3: To radarer som ligger på linjen som er parallell med satellittbanen og tilhørende hyperbler beregnet av sann TDOA pluss/minus to standardavvik.

For å begrense gale estimat før og etter radarene, må radarene ligge så langt fra hverandre at færrest mulig TDOA-er kan komme fra disse områdene. Geometrisk betyr det at færrest mulig hyperbler fra de to radarene må skjære hverandre før og etter radarene. når radarene ligger lengre fra hverandre, vil færre hyperbler skjære hverandre før og etter radarene. Da øker også sannsynligheten for at estimatoren benytter med initielle posisjoner nært en av radarene. Ettersom

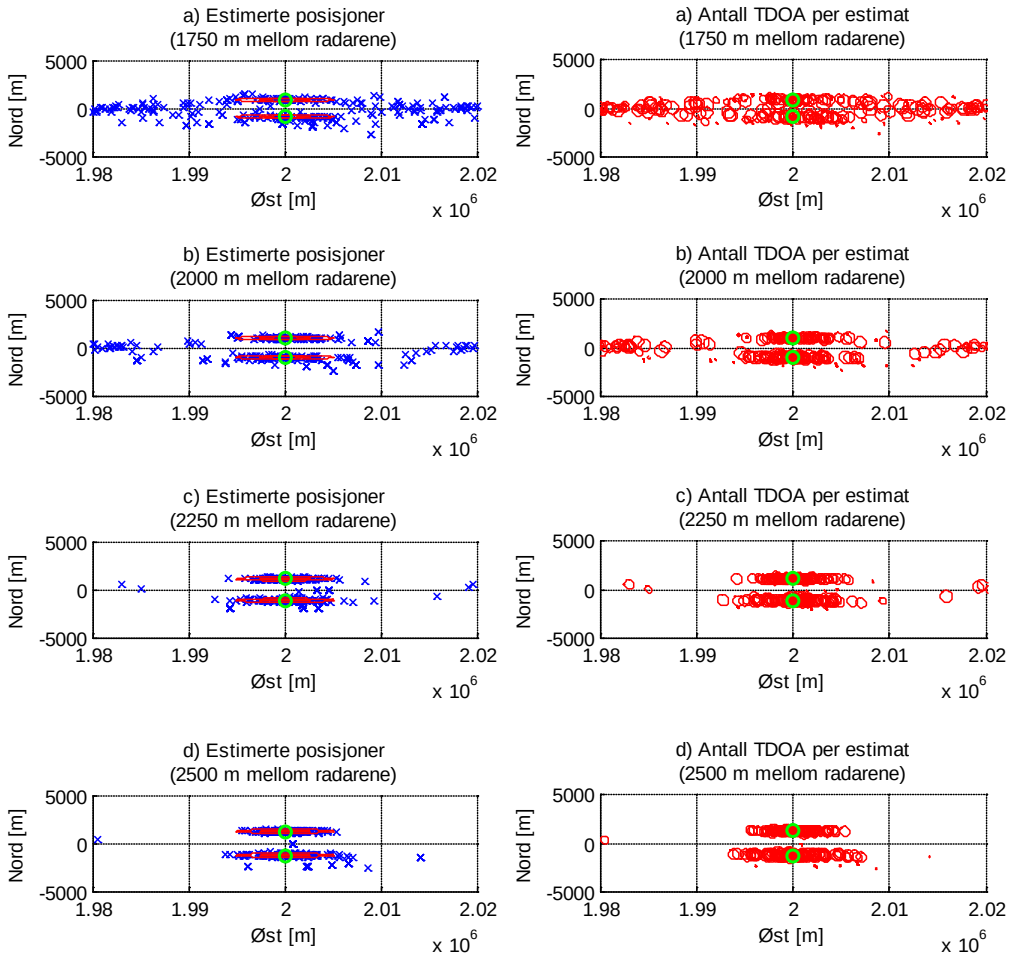
estimatoren fjerner de brukte TDOA-ene, vil områdene før og etter radarene miste mange TDOA-er som kan komme fra disse områdene, og antall gale estimat blir redusert. For å undersøke hvor langt fra hverandre radarene må ligge for at algoritmen skal gi riktige estimater for to radarer, simuleres to radarer ved ulike avstander. Figur 8.4 viser 8 figurer, hvor hver rad viser ulike avstander mellom radarene. Kolonnen til venstre viser estimerte posisjoner fra Monte Carlo simuleringer og kolonnen til høyre viser antall TDOA benyttet i estimatene. Hver Monte Carlo simulering består av 100 gjennomkjøringer.

Radaroppsett a
I radaroppsett a) er to radarer plassert med en innbyrdes avstand på 1750 m. Som figuren viser ligger det mange estimater både før og etter radarene (i retning øst). Det ser også ut til at antallet TDOA som benyttes i estimatene er like høyt som for estimatene som ligger nært radarene.

Radaroppsett b
I radaroppsett b) er to radarer plassert med en innbyrdes avstand på 2000 m. Ved å sammenlikne med radaroppsett a), er at antall gale estimat er redusert, men fortsatt inneholder de gale estimatene like mange TDOA-er som estimatene nært radarene.

Radaroppsett c
I radaroppsett c) er to radarer plassert med en innbyrdes avstand på 2250 m. Antall gale estimat er redusert betraktelig i forhold til radaroppsett b).

Radaroppsett d
I radaroppsett d) er to radarer plassert med en innbyrdes avstand på 2500 m. Algoritmen gir stort sett riktige estimater av de to radarposisjonene.



Figur 8.4: Fire tilfeller med to radarer som ligger på linjen som er parallell med satellittbanen. Kolonnen til venstre viser estimerte posisjoner (blå kryss) fra en Monte Carlo simulering, 2σ feilellipser (røde ellipser) og to radarposisjoner (grønne sirkler). Kolonnen til høyre viser antall TDOA benyttet i hvert estimat (radius lik antall TDOA multiplisert med 50 m).

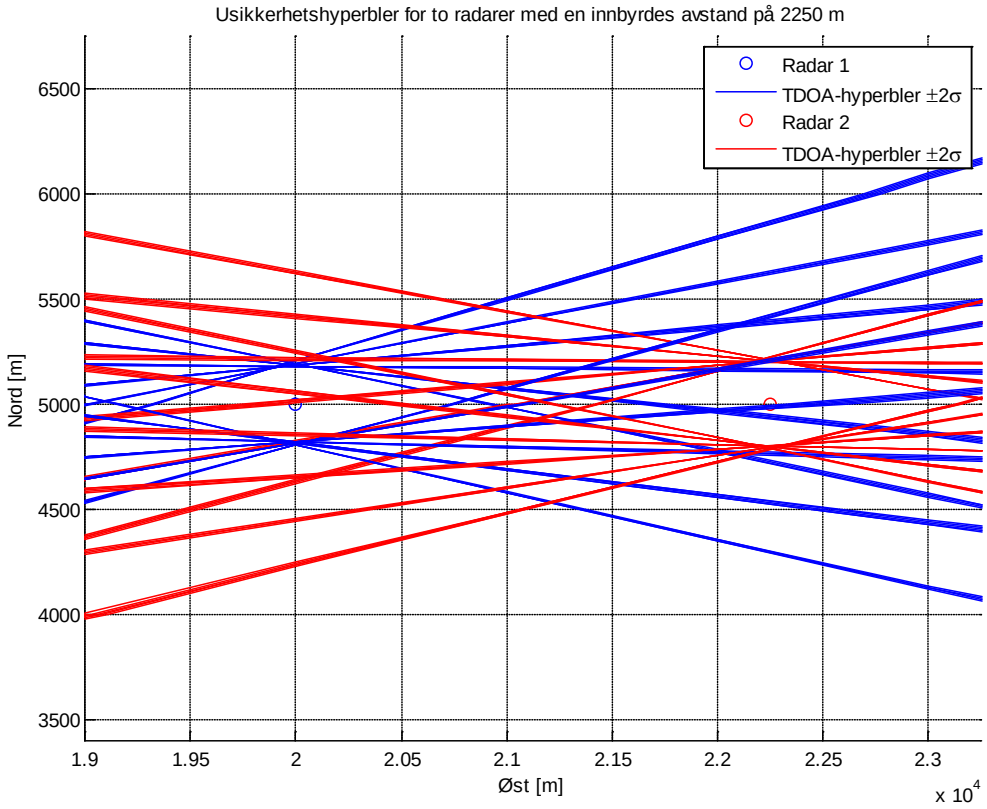
8.2 Geolokalisering av radarer i scenario med to UAS-er

Analysen av minimumsavstanden mellom to radarer i UAS-scenarioet er delt inn i to deler; avsnitt 8.2.1 hvor radarene ligger på en linje som står normalt på UAS-banen, og avsnitt 8.2.2 hvor radarene ligger på en linje som er parallell med UAS-banen. Analysene i kapittelet bygger på analysene fra avsnitt 8.1.

8.2.1 To radarer som ligger på en linje som står normalt på UAS-banen

I denne avsnittet undersøkes det hvor nært to radarer kan ligge, når de ligger på en linje som står normalt på UAS-banen, for at algoritmen allikevel skal gi riktige estimater av radarene. I avsnitt 8.1.1 måtte satellittene motta TDOA-er fra begge radarene, hvor rundt halvparten av alle TDOA-ene fra en av radarene sanne TDOA med målingsavvik (pluss/minus to standardavvik) måtte ligge utenfor den andre rada radrens sanne TDOA med målingsavvik (pluss/minus to standardavvik). Til forskjell fra satellitt-scenarioet er målingene i UAS-scenarioet utført i seks måleintervaller, og ikke i løpet av en hel overflyvning. Figur 8.5 viser to radarer som ligger på linjen i x = 5000 m,

som står normalt på flybanen til UAS-ene. Som i avsnitt 7.1, antar det at UAS-ene mottar 27 hovedlobepasseringer fra radarene, hver med 10 TDOA-er som midles. I figuren kan UAS-ene motta hovedlobepasseringer fra to måleintervall fra radar 2 (rød) hvor sann TDOA med et målingsavvik på to standardavvik ikke ligger innenfor sann TDOA med et målingsavvik på to standardavvik fra radar 1 (blå). Disse to måleintervallene inneholder totalt 9 hovedlobepasseringer, altså en tredjedel av totalt antall hovedlobepasseringer.



Figur 8.5: To radarer som ligger på linjen som står normalt på UAS-banen og tilhørende hyperbler beregnet av sann TDOA pluss/minus to standardavvik.

Figur 8.6 viser 8 figurer, hvor hver rad viser ulike avstander mellom radarene. Kolonnen til venstre viser estimerte posisjoner fra Monte Carlo simuleringer og kolonnen til høyre viser antall TDOA benyttet i estimatene. Hver Monte Carlo simulering består av 100 gjennomkjøringer.

Radaroppsett a
I radaroppsett a) er to radarer plassert med en innbyrdes avstand på 1500 m. I figuren ser det ut til at fleste parten av estimatene ligger nærmest radar 1, samtidig som det også ligger en del estimat bak radar 2. Disse har riktignok færre TDOA i forhold til estimatene nært radar 1.

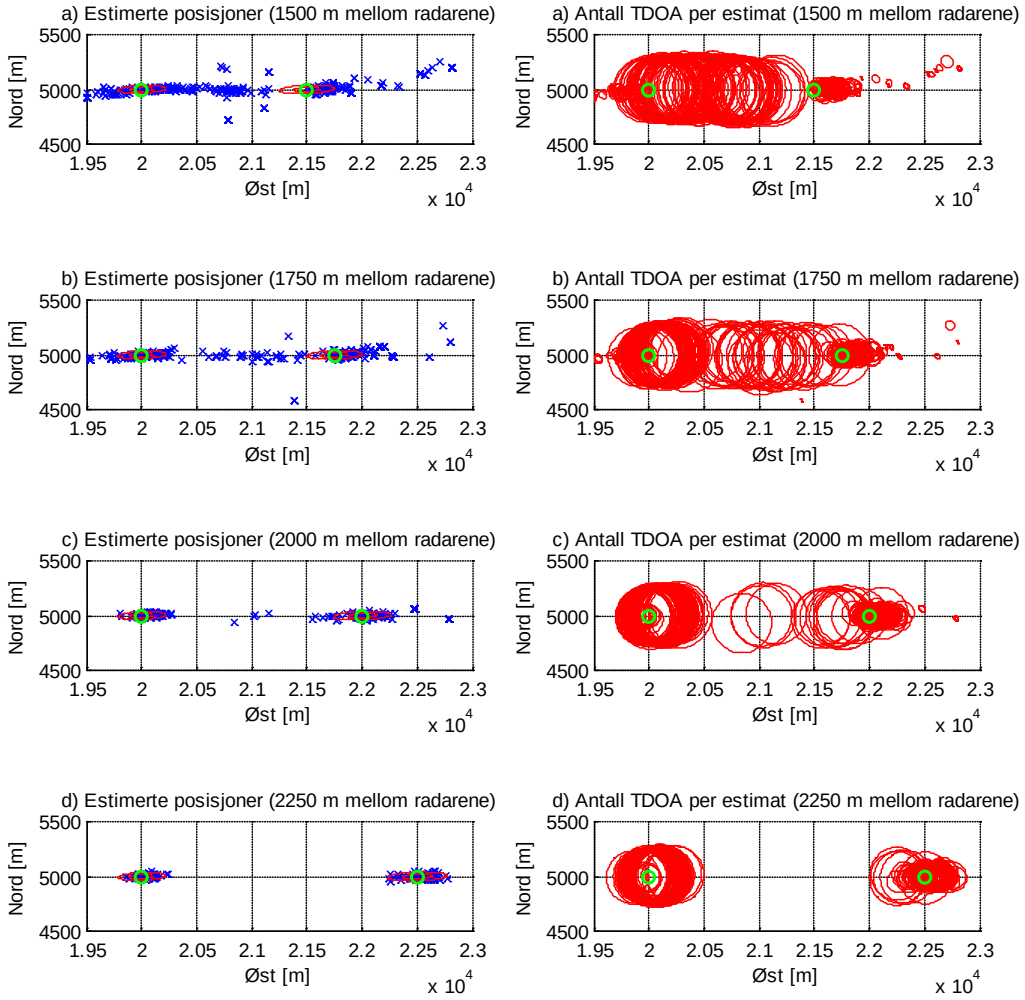
Radaroppsett b
I radaroppsett b) er to radarer plassert med en innbyrdes avstand på 1750 m. I figuren er antall gale estimat redusert i forhold til radaroppsett a), men fortsatt ligger radarene for tett til at algoritmen skal gi riktige estimater av radarposisjonene.

Radaroppsett c

I radaroppsett c) er to radarer plassert med en innbyrdes avstand på 2000 m. Radarene ligger nå så langt fra hverandre at estimatoren stort sett gir riktig estimat av hver av radarposisjonene. Allikevel gir noen av gjennomkjøringer estimater som ligger midt mellom radarene, og som har like mange TDOA som estimatene som ligger nært radarene.

Radaroppsett d

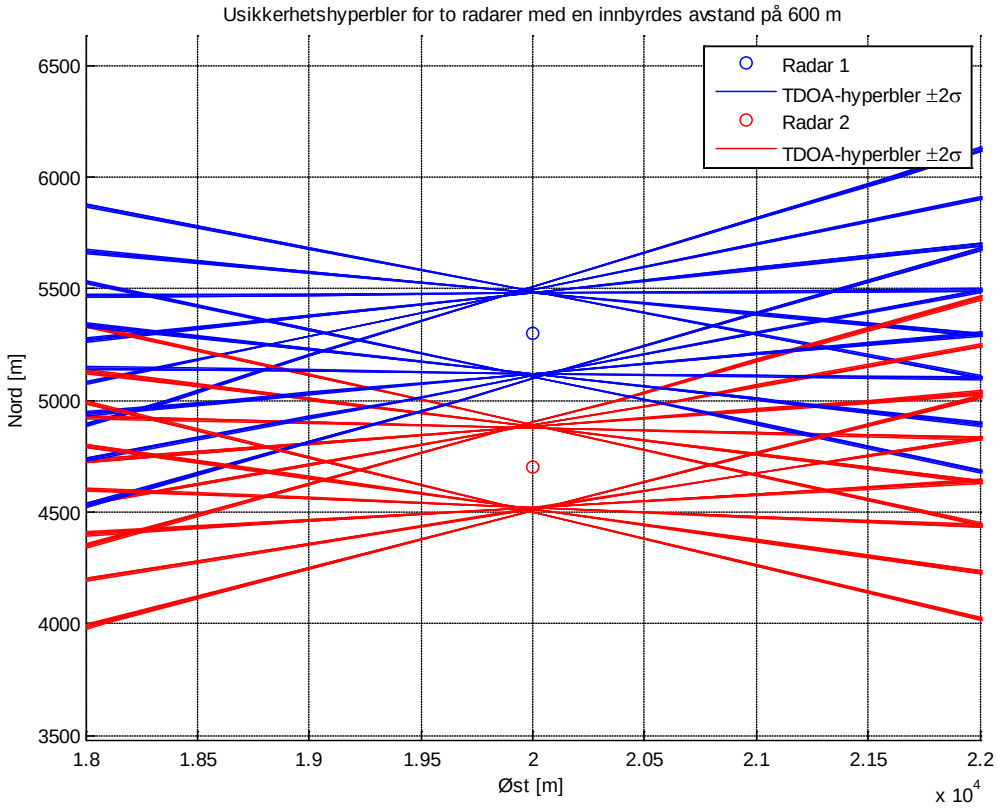
I radaroppsett d) er to radarer plassert med en innbyrdes avstand på 2250 m. Algoritmen gir i dette tilfellet bare riktige estimat som tydelig tilhører hver av radarposisjonene. Som det vises av figuren, har estimatene av radar 1 stort sett flere TDOA enn radar 2. dette tyder på at radar 1 estimeres først, og dermed benytter mange av TDOA-ene som kommer fra radar 2. Allikevel ligger radarene så langt fra hverandre at det fortsatt er nok TDOA-er til å estimere posisjonen til radar 2 etter at radar 1 er estimert.



Figur 8.6: Fire tilfeller med to radarer som ligger på linjen som står normalt på UAS-banen. Kolonnen til venstre viser estimerte posisjoner (blå kryss) fra en Monte Carlo simulering, 2σ feilellipser (røde ellipser) og to radarposisjoner (grønne sirkler). Kolonnen til høyre viser antall TDOA benyttet i hvert estimat (radius lik antall TDOA multiplisert med 5 m).

8.2.2 To radarer som ligger på en linje som er parallell med UAS-banen

Fra analysen av to radarer som ligger på en linje som er parallell med satellittbanen i avsnitt 8.1.2, fås at to radarer må ligge så langt fra hverandre at det ikke dannes områder hvor mange av TDOA-ene fra begge radarene kan komme fra. Figur 8.7 viser to radarer som ligger på en linje som står parallelt med en UAS-banen. Som i avsnitt 7.1, antas det at UAS-ene mottar 27 hovedlobepasseringer fra radarene, hver med 10 TDOA-er som midles. I figuren er det tegnet opp hyperbler for sann TDOA pluss/minus to standardavvik (for 10 midlede TDOA, σ = 22,36 ns). I figuren ligger radarene så langt fra hverandre at sann TDOA med måleusikkerhet (pluss/minus to standardavvik) fra den ene radaren ligger utenfor den andre radarens sanne TDOA med måleusikkerhet. Som i avsnitt 8.1.2 vil det kunne dannes områder både før og etter radarene (i retning øst) som mange TDOA-er kan komme fra, og som resulterer i gale estimat.



Figur 8.7: To radarer som ligger på linjen som er parallell med UAS-banen og tilhørende hyperbler beregnet av sann TDOA pluss/minus to standardavvik.

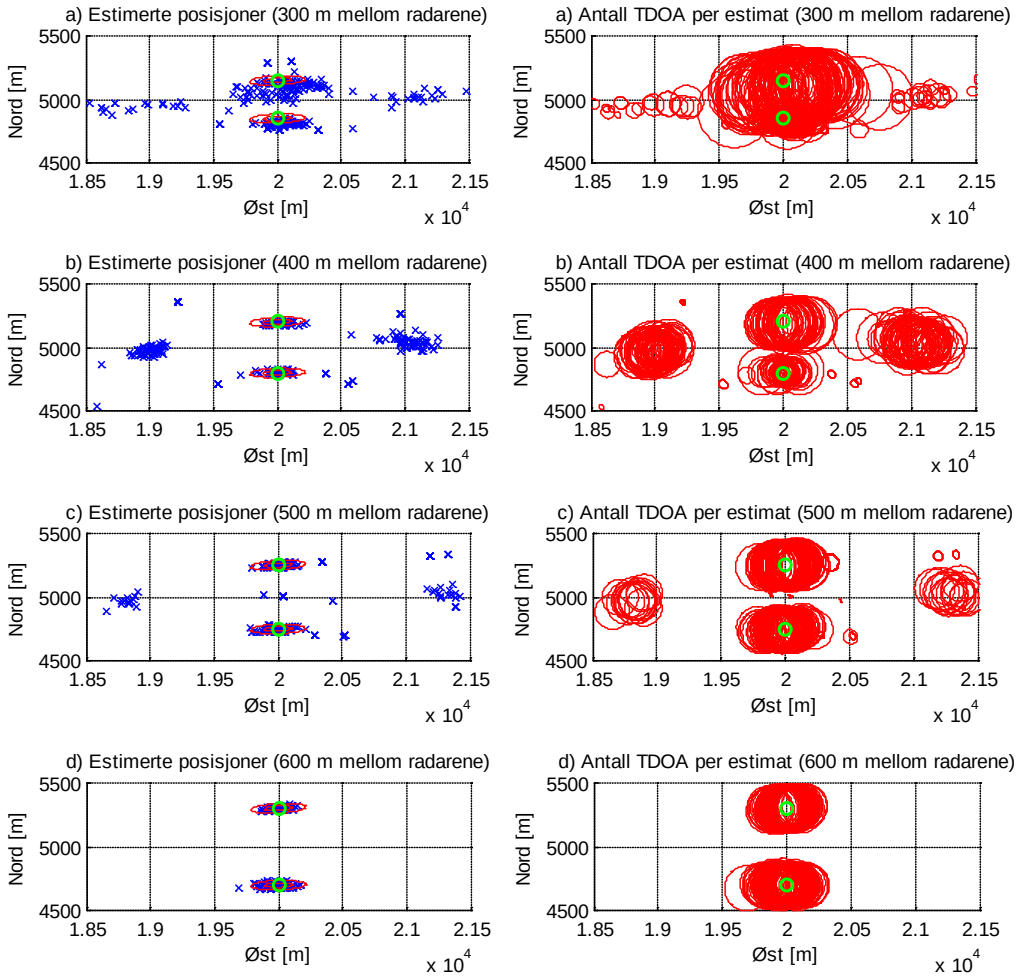
For å undersøke hvor nært to radarer kan ligge, simuleres ulike avstander mellom radarene ved Monte Carlo simulering. Figur 8.8 viser åtte figurer, hvorav hver linje med figurer gjelder ulik avstand mellom radarene. Kolonnen til venstre inneholder estimerte posisjoner ved de ulike avstandene etter 100 iterasjoner med Monte Carlo simulering. Kolonnen til høyre viser antall TDOA som er benyttet i hvert estimat.

Radaroppsett a
I radaroppsett a) er to radarer plassert med en innbyrdes avstand på 300 m. Mange av estimatene vil da ligge mellom radarene og både før og etter radarene (i retning øst). Radarene ligger for tett til at algoritmen klarer å gi riktige estimat av hver av radarene.

Radaroppsett b
I radaroppsett b) er to radarer plassert med en innbyrdes avstand på 400 m. Ved å sammenlikne med radaroppsett a), er estimatene mellom radarene forsvunnet. Samtidig ligger flere av estimatene før og etter radarene med like mange TDOA som estimatene som ligger nært radarene.

Radaroppsett c
I radaroppsett c) er to radarer plassert med en innbyrdes avstand på 500 m. Antall gale estimat før og etter radarene er redusert betraktelig, og mange av estimatene ligger nært radarene. Samtidig har de gale estimatene fortsatt like mange TDOA som de riktige estimatene.

Radaroppsett d
I radaroppsett d) er to radarer plassert med en innbyrdes avstand på 600 m. Radarene ligger nå så langt fra hverandre at algoritmen estimerer riktige estimat av begge radarene.



Figur 8.8: Fire tilfeller med to radarer som ligger på linjen som er parallell med UAS-banen. Kolonnen til venstre viser estimerte posisjoner (blå kryss) fra en Monte Carlo simulering, 2σ feilellipser (røde ellipser) og to radarposisjoner (grønne sirkler). Kolonnen til høyre viser antall TDOA benyttet i hvert estimat (radius lik antall TDOA multiplisert med 5 m).

9 Analyse av pulsassosiasjon og geolokaliseringsalgoritmen når ankomsttidene inneholder slengere

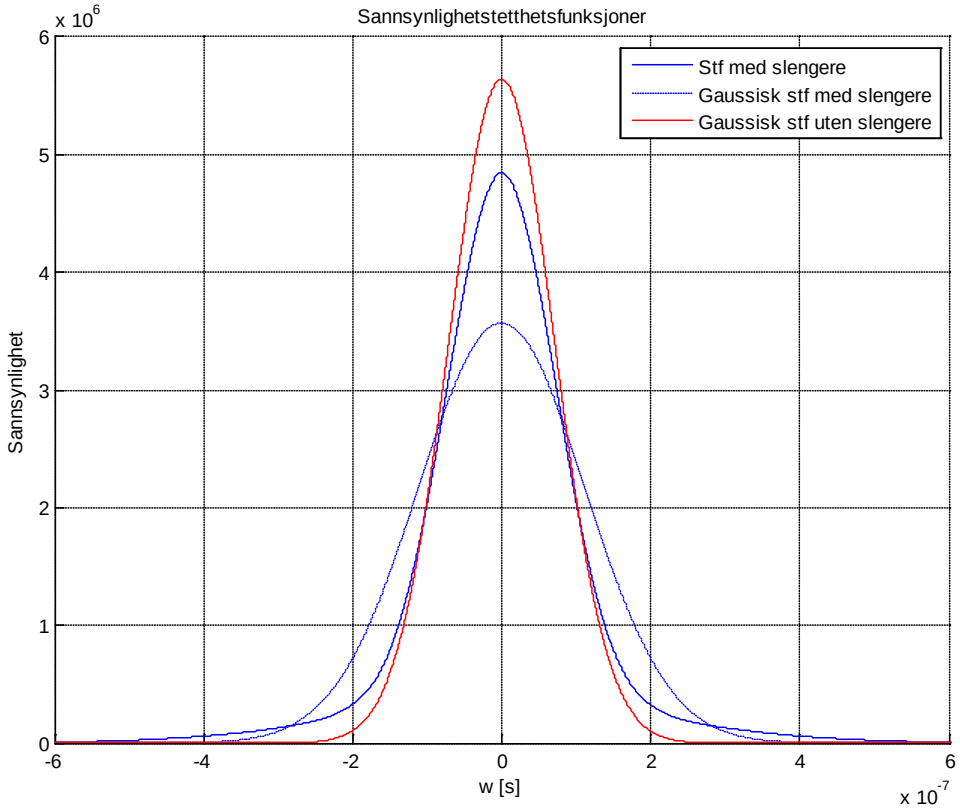
I dette kapittelet undersøkes det hvor godt pulsassosiasjon og geolokaliseringsalgoritmen fra kapittel 6 fungerer når de simulerte ankomsttidene inneholder slengere. Slengere er et fenomen som ofte oppstår i reelle målinger, og er målinger som kan se helt gale ut. Analysene i kapittelet tilsvarer noen av simuleringene som ble utført kapittel 7, og det resultatene fra analysene blir sammenliknet med tilvarende resultat når målingene er uten slengere.

Kapittelet er delt inn i tre deler. I avsnitt 9.1 analyseres det hvordan CRLB blir når målestøyen til TDOA-ene ikke lenger er gaussisk. Avsnitt 9.2 inneholder analyser for UAS-scenarioet og avsnitt 9.3 inneholder analyser satellitter-scenarioet.

9.1 Analyse av oppnåelig estimeringsnøyaktighet når ankomsttidene inneholder slengere

Målestøy, altså den additive støyen som legges på TDOA, med slengere er beskrevet i avsnitt 5.3. Fordelingen for målestøyen blir ikke-gaussisk når det inkluderes slengere i ankomsttidene. Ettersom fordelingen ikke er kjent analytisk må CRLB løses numerisk. [11] beskriver hvordan CRLB kan løses numerisk, men pga. begrenset tid for oppgaven ble dette ikke prioritert. Fordelingen for målestøy med slengere ble isteden erstattet med en gaussisk fordeling med samme middelverdi og standardavvik (σ = 112 ns) som den ikke-gaussiske fordelingen. Denne gaussiske antakelsen ble deretter benyttet i CRLB.

Figur 9.1 viser den numeriske fordelingen for målestøy med slengere (blå kurve) og gaussisk fordeling for målestøy uten slengere (rød kurve). I tillegg er det tegnet inn en gaussisk fordeling med samme middelverdi og standardavvik som den numeriske fordelingen for målestøy med slengere (blå stiplet kurve). Som figuren viser er den gaussiske fordelingen med slengere noe bredere enn den ikke-gaussiske fordelingen med slengere, bortsett fra støy langt ute på hver side. Det betyr at spredningen til TDOA trukket fra den gaussiske fordelingen med slengere sannsynligvis er større enn for den ikke-gaussiske fordelingen, og at CRLB med gaussisk antakelse gir en nedre grense for estimeringsnøyaktigheten som er større enn hva CRLB den ikke-gaussiske fordeling gir.



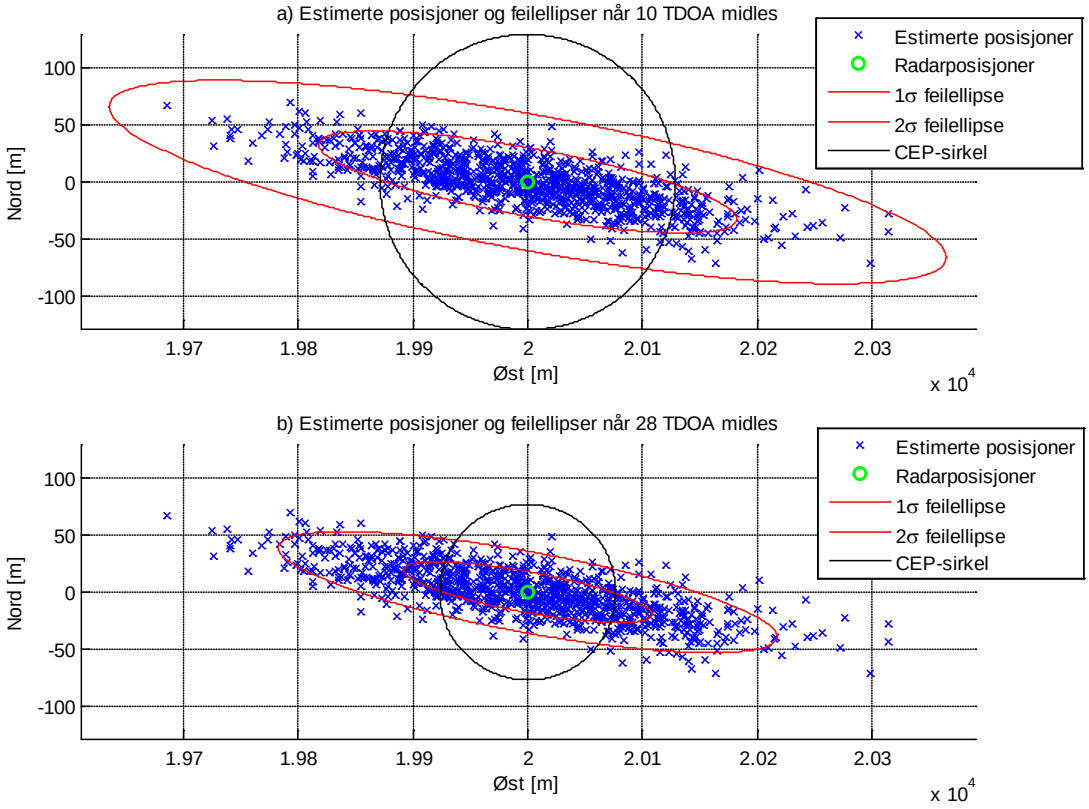
Figur 9.1: Ikke-gaussisk fordeling for målestøy med slengere (blå kurve), gaussisk tilnærming av fordeling med målestøy (stiplet blå kurve) og gaussisk fordeling for målestøy uten slengere (rød kurve).

9.2 Geolokalisering av radarer i UAS-scenarioet når ankomsttidene inneholder slengere

9.2.1 En radar i UAS-scenarioet

Som i avsnitt 7.1.2 undersøkes det hvor godt algoritmen estimerer radarposisjoner når det ikke er noen tvil om hvilke pulser som hører sammen i de to sensorene og hvilke pulser som kommer fra samme radar. Nytt her er at ankomsttidene inneholder slengere. Som beskrevet i avsnitt 5.3, består støyen på ankomsttidene ved at 10 % er trukket fra en gaussisk fordeling med middelverdi 0 ns og standardavvik 200 ns og de resterende 90 % fra en gaussisk fordeling med middelverdi 0 ns og standardavvik 50 ns. De 10 % slengerne er tilfeldig fordelt utover ankomsttidene. Dette fører til at rundt 20 % av TDOA-ene inneholder slengere. For å undersøke estimeringsnøyaktigheten benyttes CRLB. I avsnitt 9.1 ble den ikke-gaussiske fordelingen for målestøy med slengere erstattet av en gaussisk fordeling med middelverdi 0 ns og standardavvik 112 ns. CRLB er dermed beregnet på bakgrunn av at TDOA-ene er belagt med gaussisk målestøy med slengere. Resten av antakelsene i CRLB er identiske med antakelsene i avsnitt 7.1.2. Figur 9.2 a) viser estimerte posisjoner ved 1000 gjennomkjøringer av Monte Carlo simulering, og tilhørende feilellipser beregnet ved CRLB. I figuren ligger 83,9 % av estimatene innenfor 1σ feilellipsen, 100 % innenfor 2σ feilellipsen og 77,9 % innenfor CEP-sirkelen. Andelen estimat innenfor de ulike feilellipsene ligger altfor høyt, og CRLB gir helt tydelig gal kovariansmatrise. I

simuleringen mottar UAS-ene i snitt 28 pulser per hovedlobepassering som det kan beregnes TDOA mellom. Figur b) viser samme simulering som i figur a), men hvor CRLB er beregnet på nytt, med 28 TDOA per midlede TDOA. Da ligger 45,6 % av estimatene innenfor 1σ feilellipsen, 91,7 % innenfor 2σ feilellipsen og 52,1 % innenfor CEP-sirkelen. Antall estimat innenfor feilellipsene er fortsatt høyere enn det teoretiske. Figur 9.1 viste at den gaussiske tilnærmingen av fordelingen for målestøy er noe bredere enn den ikke-gaussiske fordelingen for målestøy. Det kan derfor tyde på at CRLB med den gaussiske antakelsen trolig gir større kovariansmatrise enn for den ikke-gaussiske fordelingen, og dermed ligger for mange estimat innenfor feilellipsene.



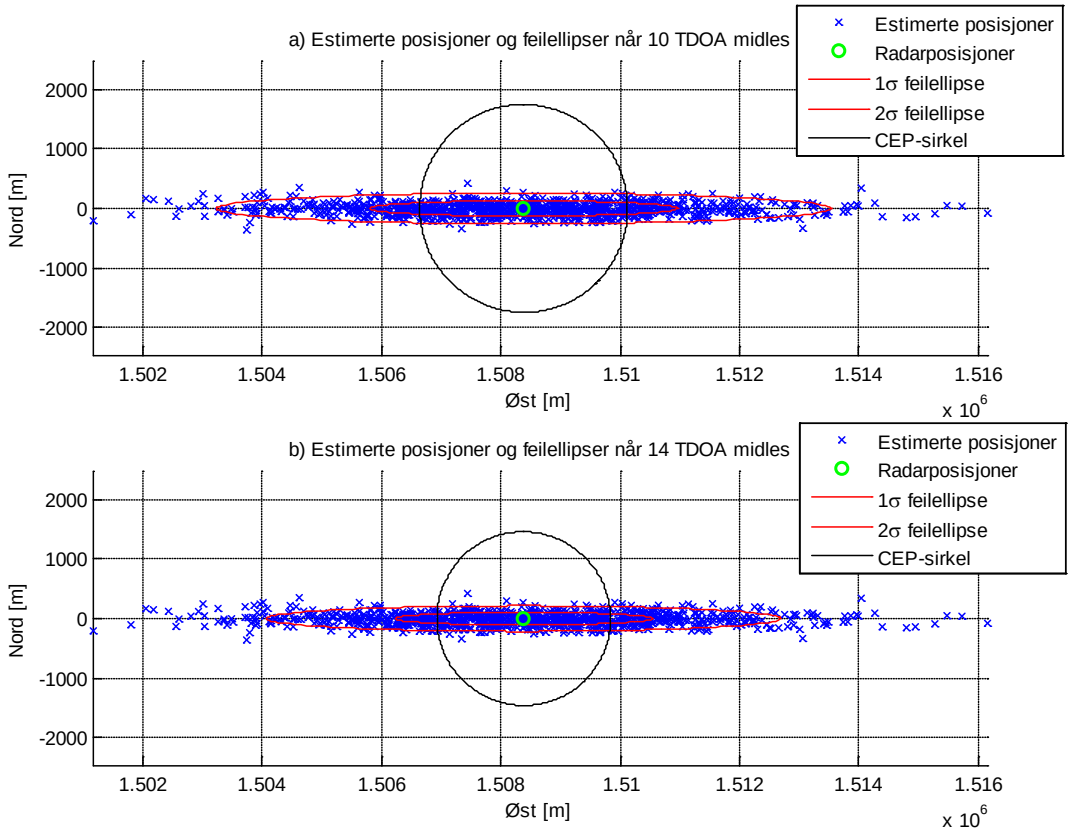
Figur 9.2: Estimerte posisjoner av en radar i UAS-scenarioet når ankomsttidene inneholder slengere fra 1000 gjennomkjøringer av Monte Carlo simulering, 1σ og 2σ feilellipser og CEP-sirkel. I a) er 10 TDOA midlet per hovedlobepassering, og i b) 28 TDOA midlet per hovedlobepassering.

9.3 Geolokalisering av radarer i satellitt-scenario når ankomsttidene inneholder slengere

9.3.1 En radar i satellitt-scenario

Som i avsnitt 7.2.2 undersøkes det hvor godt algoritmen estimerer radarposisjoner når det ikke er noen tvil om hvilke pulser som hører sammen i de to sensorene og hvilke pulser som kommer fra samme radar. Nytt her er at ankomsttidene inneholder slengere. Til å undersøke estimeringsnøyaktigheten benyttes CRLB ved samme antakelser i avsnitt 7.2.1, bortsett fra at TDOA-ene inneholder slengere. Fordelingen for målestøy med slengere antas å være gaussisk, og defineres i

avsnitt 9.1. Figur 9.3 a) viser estimerte posisjoner ved en Monte Carlo simulering med 1000 gjennomkjøringer. I figuren er det tegnet opp 1σ og 2σ feilellipser samt CEP-sirkel for posisjonsusikkerhet vha. CRLB. I figuren ligger 53,3 % av estimatene innenfor 1σ ellipsen, 93,1 % innenfor 2σ ellipsen og 56,8 % innenfor CEP-sirkelen. I CRLB antas det at 10 TDOA midles per hovedlobepassering. I simuleringen mottar satellittene i snitt 14 pulser per hovedlobepassering som det kan beregnes TDOA mellom. Figur b) viser samme simulering som figur a), men hvor CRLB er beregnet med 14 TDOA per midlet TDOA. Da ligger 40,0 % av estimatene ligger innenfor 1σ ellipsen, 85,7 % innenfor 2σ og 48,4 % innenfor CEP-sirkelen. Andelen estimater innenfor feilellipsene er fortsatt høyere enn det teoretiske. Og som i avsnitt 7.2.2 kan dette tyde på at CRLB med den gaussiske antakelsen trolig gir større kovariansmatrise enn for den ikke-gaussiske fordelingen, og dermed ligger for mange estimat innenfor feilellipsene.



Figur 9.3: Estimerte posisjoner av en radar i satellitt-scenarioet når ankomsttidene inneholder slengere fra 1000 gjennomkjøringer av Monte Carlo simulering, 1σ og 2σ feilellipser og CEP-sirkel. I a) er 10 TDOA midlet per hovedlobepassering, og i b) 14 TDOA midlet per hovedlobepassering.

10 Diskusjon av analyser

Den egenutviklede algoritmen ble i kapitlene 7, 8 og 9 testet på ulike radarformasjoner i den hensikt å finne forhold som avgjør hvor mange samtidige radarer som lar seg geolokalisere. Gjennom analysene kom det frem noen fordeler og ulemper ved algoritmen, og noen typiske situasjoner hvor algoritmen ikke gir riktige estimater av radarene.

I analysene i oppgaven er det kun benyttet simulerte ankomsttider til pulser generert av simulatoren beskrevet i kapittel 4. Fordelen med å benytte simulerte ankomsttider er at all informasjon om sensorplattformene og radarene er kjent. I en simuleringssituasjon vil støyen være kontrollert, slik at støyen kan trekkes fra ønskede fordelinger. Dette gjør det enkelt å kjøre Monte Carlo simuleringer, hvor støyen trekkes på nytt i hver gjennomkjøring. I tillegg har det da vært mulig å analysere algoritmen på målinger fra det samme radaroppsett med og uten slengere. Algoritmen burde også blitt testet på reelle målinger, ettersom det kan være fenomener som påvirker resultatene som ikke er modellert. I et reelt måleoppsett vil f.eks. praktiske begrensninger kunne føre til at sensorene mister enkle pulser fra radarene, eller til og med hele hovedlobepasseringer. Dette vil redusere antall målinger, og radarene kan dermed bli vanskeligere å geolokalisere. Datasettene ved FFI med måleopptak med ankomsttider til to plattformer hadde for dårlig tidsnøyaktighet for denne anvendelsen, og kunne dermed ikke brukes.

Dette kapittelet inneholder utvalgte problemstillinger ved resultatene fra kapitlene 7, 8 og 9, og er delt inn i fem avsnitt; i avsnitt 10.1 diskuteres estimeringsnøyaktigheten til algoritmen når sensorene mottar pulser fra en radar. Avsnitt 10.2 diskuterer hvor godt algoritmen geolokaliserer flere radarer og hva som typisk skiller riktige fra gale estimater, mens i avsnitt 10.3 diskuteres noen ulike radarformasjoner som algoritmen har problemer med. Avsnitt 10.4 oppsummerer positive og negative erfaringer ved algoritmevalget og i avsnitt 10.5 foreslås det videre arbeid innenfor problemstillingen.

10.1 Estimeringsnøyaktighet til estimatoren

Dette avsnittet diskuter estimeringsnøyaktigheten til estimatoren, og baserer seg på resultater fra analysene i avsnitt 7.1.2 og 7.2.2 hvor målingene var belagt med gaussisk målestøy og fra kapittel 9 hvor målingene var belagt med ikke-gaussisk målestøy. Diskusjonen er delt inn i to deler; en for UAS-scenarioet og en for satellitt-scenarioet.

Estimeringsnøyaktighet i UAS-scenarioet
Algoritmen ble i avsnitt 7.1 testet på simulerte ankomsttider i UAS-scenarioet. Der ble algoritmen først benyttet på ankomsttider generert av en radar. Estimeringsnøyaktigheten til algoritmen kan da undersøkes når det ikke er noen tvil om hvilke pulser som hører sammen i de to sensorene og hvilke pulser som kommer fra samme radar. I analysen var estimerinsnøyaktigheten dårligere enn hva som er teoretisk oppnåelig i følge CRLB (det ble antatt 30 TDOA per hovedlobepassering som ble midlet). Dette skylles at estimatoren velger ut TDOA som ligger innenfor pluss/minus to standardavvik (for midlede TDOA) fra sann TDOA fra den initielle posisjonen (og senere de estimerte posisjonene i estimeringsalgoritmen). Dette er nødvendig for å skille TDOA fra ulike radarer fra hverandre, men fører samtidig til at estimatoren ikke benytter all tilgjengelig informasjon (benytter teoretisk «bare» 95 % av TDOA-ene). I avsnitt 9.2 ble algoritmen testet på de samme ankomsttidene, men hvor det var lagt på slengere på 10 % av ankomsttidene i de to

sensorene. Fordelingen for TDOA ble da ikke-gaussisk, og CRLB må derfor løses numerisk. Grunnet begrenset tid for oppgaven, ble det antatt at fordelingen for målestøyen på TDOA-ene var gaussisk i CRLB (med samme middelverdi og standardavvik som den ikke gaussiske fordelingen). I analysen ble estimeringsnøyaktigheten bedre enn hva som er teoretisk oppnåelig ifølge CRLB (det ble antatt 28 TDOA per hovedlobepassering som ble midlet). Dette skylles trolig at CRLB med den gaussiske antakelsen gir en teoretisk estimeringsnøyaktighet som er dårligere enn hva CRLB for den ikke-gaussiske fordelingen gir. Begge simuleringene viser at estimatoren er forventningsrett, og at spredningen til estimatene ligger i nærheten av hva CRLB tilsier. Det er verdt å merke at gjennomsnittet av antall TDOA per hovedlobepassering er 30 når ankomsttidene er uten slengere, men 28 når ankomsttidene er med slengere. Dette viser at klyngeanalysen for å assosiere pulser i de to sensorene fra avsnitt 6.1 utelater i gjennomsnitt to TDOA-er fra hver hovedlobepassering med TDOA-er, når TDOA-ene inneholder slengere. I [2] ble det utført en slenger-test for å fjerne slengere fra TDOA-ene som ble midlet, men ved å benytte klyngeanalyse til å assosiere TDOA fra samme radar fjernes slengerne automatisk.

Estimeringsnøyaktighet i satellitt-scenarioet.
I avsnitt 7.2 ble algoritmen testet på simulerte ankomsttider fra UAS-scenarioet. Også her algoritmen først benyttet på ankomsttider generert av en radar. Som for scenario med to UAS-er, ble estimeringsnøyaktigheten dårligere enn hva som er teoretisk oppnåelig ifølge CRLB (det ble antatt 15 TDOA per hovedlobepassering som ble midlet). Allikevel viste simuleringen at estimatoren er forventningsrett, og at spredningen til estimatene følger CRLB ganske godt. I avsnitt 9.3 ble det introdusert ankomsttider med slengere. I CRLB ble det da antatt at hver hovedlobepassering resulterer i 14 TDOA-er som midles. Som for UAS-scenarioet ble det antatt en gaussisk tilnærming av fordelingen for målestøyen på TDOA. Også her var estimeringsnøyaktigheten mye bedre enn hva som er oppnåelig ifølge CRLB, og CRLB med den gaussiske tilnærmingen er dermed feil.

10.2 Geolokalisering av radarer

Dette avsnittet diskuterer hvor godt algoritmen geolokaliserer radarer når sensorene mottar pulser fra flere samtidige radarer og hva som skiller riktige estimater fra gale estimater. Analysen baserer seg på resultater fra analysene i kapittel 7. Diskusjonen er delt inn i to deler; en for scenario med to UAS-er og en for scenario med to satellitter.

Geolokalisering av radarer i UAS-scenarioet
I avsnittene 7.1.3 og 7.1.4 ble algoritmen testet på simulerte ankomsttider fra hhv. 6 og 24 radarer. Disse analysene ble kun utført på ankomsttider uten slengere. Når sensorene mottok pulser fra seks radarer, ble alle radarposisjonene estimert i alle Monte Carlo gjennomkjøringene. Estimatene av radarposisjonene skilte seg tydelig fra gale estimater ved at antall TDOA som ble benyttet i de riktige estimatene ofte kunne være oppimot 5 ganger høyere enn for de gale estimatene. I analysene ble det antatt at sensorene mottok 10 pulser per hovedlobepassering som ble midlet. Grunnen til at 10 TDOA per midlet TDOA ble valgt, var at antallet typisk lå mellom 10 og 40. Ved å velge 10 TDOA per midlet TDOA fås en CRLB som gir en høyere usikkerhet enn hva spredningene til estimatene faktisk er, og andelen estimat innenfor de ulike feilellipsene blir høyere enn hva som er oppnåelig ifølge CRLB. I simuleringen lå estimatene til de ulike radarene sentrert om radarposisjonene, og spredningen stemte forholdsvis godt med CRLB (selv om for mange av estimatene lå innenfor feilellipsene). Simuleringen med 24 radarer ble utført for å undersøke hvordan algoritmen takler pulser fra mange samtidige radarer og hvilke radarformasjo-

ner som gav problemer for algoritmen. I Monte Carlo simuleringen ble 19 av radarposisjonene estimert i hver gjennomkjøring, mens de resterende 5 radarposisjonene ble estimert med estimater som kan tilhøre flere radarer eller som ikke ble estimert i det hele tatt. Situasjonen med disse fem radarene diskuteres nærmere i avsnitt 10.3.

Geolokalisering av radarer i satellitt-scenarioet.
Algoritmen ble i avsnitt 7.2.3 testet på simulerte ankomsttider fra 10 radarer spredd utover hele det opplyste området, i avsnitt 7.2.4 testet på 10 radarer plassert tett i senter av det opplyste området og i avsnitt 7.2.5 testet på 50 radarer spredd utover hele det opplyste området. Som for UAS-scenarioet ble disse analysene kun utført på ankomsttider uten slengere. Når 10 radarer var spredd utover hele det opplyste området, lå radarene så langt fra hverandre at algoritmen estimerte riktige estimater av radarposisjonene. Når 10 radarer lå tett i senter av hovedloben, var det to radarer som vær særlig interessante. Radar 1 og radar 7 lå på linje, sett fra satellittene, og algoritmen gav kun felles estimater av radarposisjonene. Oppsettet med 50 radarer ble simulert av den hensikt å se hvor mange samtidige radarer som algoritmen klarer å geolokalisere. I analysen ble 41 av de 50 radarposisjonene estimert med riktige estimat i hver gjennomkjøring. Som i tidligere analyser gir algoritmen estimater ikke riktige estimater av radarer når disse ligger tett sammen på linje, sett fra satellittene. I analysene for satellitt-scenariet ble det også antatt i CRLB at satellittene mottok 10 pulser per hovedlobepassering som det ble beregnet TDOA mellom, og at disse 10 TDOA-ene ble midlet. Som for UAS-scenarioet var antallet TDOA per midlet TDOA litt høyere i simuleringene, enn hva som ble antatt i CRLB. Dette førte til at CRLB gav litt for store feilellipser og at det dermed lå for mange estimater innenfor de ulike ellipsene. Allikevel viste simuleringene at de riktige estimatene lå sentrert rundt radarposisjonene med forholdsvis lik spredning som CRLB tilsier.

10.3 Typiske radarformasjoner som ikke lar seg geolokalisere

I analysene er det noen radarformasjoner som går igjen hvor algoritmen gir et estimat for flere radarer, hvor bare en av radarposisjonene blir estimert, eller hvor det dannes mange gale estimater. Generelt får algoritmen problemer med å gi riktige estimat av radarposisjoner når radarene ligger så tett at TDOA-ene fra radarene er nesten like, eller at radarene ligger på linje, sett fra sensorene, slik at mange av TDOA-ene fra den ene radaren også kan komme fra den andre radaren.

Avsnitt 7.1.4 inneholder en analyse hvor to UAS-er mottar ankomsttider fra 24 radarer. Analysen inneholder begge nevnte problemer. Førstnevnte problem er tilfellet for radar 14, 15 og 16. Her ligger radarene rundt 350 m fra hverandre, og TDOA-ene fra de tre radarene er nesten identiske. Når en av radarposisjonene estimeres benyttes nesten alle TDOA-ene fra de to andre radarene. Resultatet blir da et estimat for de tre radarene med nesten tre ganger så mange TDOA-er som de andre estimatene i simuleringen. Det vil ikke være mulig å skille pulsene fra disse tre radarene ved å kun se på TDOA. For å få tre estimater til disse tre radarene, må pulsene fra de ulike radarene skilles ved andre metoder som f.eks. rundetid og PRI. Videre lå radar 20, 21 og 22 slik til at nesten alle TDOA-er fra radar 21 og 22 passer sammen med TDOA-er fra radar 20. Dermed fikk radar 20 flest TDOA-er som lå i nærheten, og ble dermed estimert først. Etter at radarposisjonen var estimert, ble mange av TDOA-ene som tilhørte radar 21og 22 fjernet. De gjenværende TDOA-ene fra radar 21og 22 dannet da nye estimater med mange TDOA-er som ikke lå i

nærheten av noen radar. Denne situasjonen burde ikke løses ved greedy-filosofien, ettersom radar 20 stjeler mange av TDOA-ene som tilhører de to andre radarene. Her kan det tenkes at metoden fra [2] hadde passet bedre, hvor algoritmen estimerer posisjoner fra alle de initielle posisjonene, uten å fjerne TDOA-er underveis. Forhåpentligvis ville de tre radarposisjonene stått frem ved at mange initielle posisjonene gav estimater nært disse tre radarposisjonene. Utfordringen da ville vært å slå sammen estimatene som er estimater av samme radarposisjon, og hvordan TDOA-ene skal fordeles mellom de ulike estimatene (TDOA-ene har en radar de kommer fra, og bør dermed ikke benyttes til å estimere flere radarposisjoner).

I kapittel 8 ble det analysert hvor nært to radarer kan ligge for at algoritmen allikevel skal gi estimater av begge radarene. I analysene ble det kun analysert to radarer som lå på en linje som står normalt på sensorbanen og parallelt med sensorbanen. I satellitt-scenarioet ses det kun på hele overflyvninger, så den minste avstanden mellom radarene er bare avhengig av hvor i det opplyste området radarene ligger. For UAS-scenarioet derimot er det i oppgaven kun sett på seks måleintervall, og radarenes posisjon i forhold til UAS-ene har derfor en betydning. Analysene viste at for to radarer som ligger på en linje som står normalt på sensorbanen er det avgjørende at radarene ligger så langt fra hverandre at sensorene mottar mange nok TODA-er fra hver av radarene som ikke overlapper. Det vil si at etter at posisjonen til den ene radaren er estimert, er det fortsatt nok TDOA-er igjen til å estimere den andre radaren. I analysene som er utført er det ikke funnet noe fast antall TDOA-er som sensorene må motta fra hver av radarene som ikke overlappes med den andre radarens TDOA, men det ser ut til at minimum 30-50 % av TDOA-ene fra hver radar ikke bør overlappe. Videre viste analysene at dersom to radarer ligger på en linje som er parallell med sensorbanen, er det avgjørende at radarene ligger så langt fra hverandre at det ikke dannes områder hvor mange hyperbler fra begge radarene skjærer hverandre og dermed danner gale estimater. Her er det avgjørende at estimatoren velger å begynne å estimere posisjoner til radarene først. Ettersom de brukte TDOA-ene fjernes, vil det ikke være noen TDOA-er igjen til å estimere posisjoner i områdene hvor mange av hyperblene fra begge radarene skjærer hverandre. Selv om det i analysene i kapittel 8 kun ble analysert hvor nært to radarer kunne ligge når de enten lå på en line som ligger normalt med sensorbanen og parallelt med sensorbanen, vil prinsippene som ble funnet i analysen være gjeldende for alle radarer som ligger i nærheten av hverandre. For at algoritmen skal gi riktige estimater av radarposisjonene, må radarene ligge slik til at sensorene mottar nok TDOA-er fra hver av radarene som ikke benyttes av estimatet av den andre radaren. Og tilsvarende må radarene ligge slik til at det ikke er områder hvor mange av TDOA-ene fra begge radarene passer kan komme fra, slik at det estimeres et galt estimat i dette området.

Avsnitt 7.2.4 inneholder en analyse hvor to satellitter mottar ankomsttider fra 10 radarer som er plassert tett midt i det opplyste området. Her ligger radar 1 og 7 på en linje som står tilnærmet normalt på satellittbanen. Algoritmen gir da stort sett estimater som ligger mellom radarene. Ved å sammenlikne radar 1 og 7 med analysen for hvor tett radarene kan ligge i avsnitt 8.1.1, ses det at radar 1 og 7 ligger for tett (omtrent 20 km fra hverandre). Avstanden mellom dem må minst være 50 km for at algoritmen skal gi estimater av begge radarposisjonene. Det kan tenkes at metoden nevnt i avsnittet ovenfor som omhandler radar 20, 21 og 22 ville passet bedre her.

10.4 Algoritmeaspekter

Algoritmen ble basert på en greedy-filosofi får å forhindre at mange av de initielle posisjonene gav samme estimater, for å redusere antall gale estimater, og for å begrense regnetiden ved at antall estimat reduseres. Dette betyr at algoritmen tilordner TDOA til det estimatet som ser mest lovende ut der og da, selv om dette ikke nødvendigvis er den beste løsningen totalt sett. I algoritmen fås stort sett kun et estimat for hver radar, og det er lett å sortere de riktige estimatene fra de gale estimatene. Allikevel fører algoritmen med seg noen ulemper. I avsnitt 10.3 diskuteres det noen tilfeller hvor greedy-filosofien trolig ikke gir den beste løsningen. Dette er tilfeller hvor to radarer ligger slik at TDOA-ene fra den ene av radarene passer godt til å komme fra den andre radaren. I disse tilfellene kan det tenkes at metoden fra [2] hadde passet bedre, hvor posisjonene ble estimert uten at brukte TDOA-er ble fjernet. Derimot fungerer algoritmen godt når radaren ligger slik at sensorene mottar mange TDOA-er som kun kan komme fra radaren som skal estimeres. I avsnitt 7.2.5 ble algoritmen testet på ankomsttider fra 50 radar hvor algoritmen assosierte 43 av radarene i hver gjennomkjøring av Monte Carlo simuleringen. Dette tyder på at algoritmen assosierer TDOA-ene fra de ulike radarene, selv ved et komplisert signalmiljø.

Ytelsen til algoritmen er dårlig når det gjelder tidsbruk. Appendiks F inneholder to tabeller hvor det er listet opp antall ankomsttider, antall TDOA, gjennomsnittlig tidsbruk for gjennomkjøringene av Monte Carlo simuleringene, osv. for analysene som er utført. Algoritmen brukte rundt ett sekund per Monte Carlo gjennomkjøring når algoritmen ble testet på pulser fra en radar. Når algoritmen ble benyttet på 10 radarer spredd utover hele hovedloben i satellitt-scenarioet benyttet algoritmen 35 sekunder per Monte Carlo gjennomkjøring, mens når radarene lå tett i senter av hovedloben i satellitt-scenarioet benyttet algoritmen 216 s. Algoritmen ble også testet i to tilfeller med mange radarer. For testen med 24 radarer i UAS-scenarioet benyttet algoritmen rundt 24 minutter per Monte Carlo gjennomkjøring og når algoritmen ble testet på 50 radarer i satellitt-scenarioet benyttet algoritmen rundt 4 timer Monte Carlo gjennomkjøring. Tidsbruket skylles hovedsakelig klyngeanalysen når de initielle posisjonene skal bestemmes og estimeringen av posisjonene. Det er vanskelig å finne en fast regel for hvor lang tid algoritmen bruker på å geolokalisere et antall radarer. F.eks. benytter en gjennomkjøring av testen med 24 radarer i UAS-scenarioet 24 minutter. I simuleringen mottar UAS-ene i overkant av 33 000 ankomsttider i de to sensorene, noe som gir 895 midlede TDOA og 180 402 skjæringspunkter mellom hyperblene beregnet av de midlede TDOA-ene. Skjæringspunktene resulterer i 2810 initielle posisjoner og til slutt 45 estimerte posisjoner. Til sammenlikning tar en gjennomkjøring av simuleringen med 50 radarer i satellitt-scenarioet 4 timer. I simuleringen mottar satellittene-ene i overkant av 22 000 ankomsttider i de to sensorene, noe som gir 1396 midlede TDOA og 68 258 skjæringspunkter mellom hyperblene (kun innenfor området som hovedlobene til begge satellittene dekker) beregnet av de midlede TDOA-ene. Skjæringspunktene resulterer i 56 054 initielle posisjoner og til slutt 123 estimerte posisjoner. Det er ingen klar sammenheng mellom antall midlede TDOA og tidsbruk, bortsett fra at algoritmen bruker lengre tid når den mottar flere pulser fra flere radarer. Uansett bruker klyngeanalysen mye av tiden. Det kan derfor tenkes at et rektangulært grid som ble benyttet i [2] hadde kortet ned beregningstiden, men da ville algoritmens evne til å plassere initielle posisjoner i de områder med mange hyperbler blitt tapt.

10.5 Videre arbeid

I oppgaven er det kun analysert tilfeller hvor UAS-ene mottar pulser fra seks måleintervaller. Det er essensielt at sensorene mottar TDOA-er fra radarene i ulike sensorposisjoner for at radarene skal la seg geolokalisere. I oppgaven ble det begynt på en analyse av hvordan radarene lar seg geolokalisere når sensorene mottar TDOA-er fra færre måleopptak, men ble analysen ble ikke fullført på grunn av begrenset tid i oppgaven. Foreløpig analyse er beskrevet i appendiks G.1.

Analysene har vist flere situasjoner hvor algoritmen ikke klarer å skille to radarer. Typisk i disse tilfellene er radarene plassert slik at TDOA-ene fra den ene radaren overlapper TDOA-ene til den andre radaren. Ved disse situasjonene fungerer greedy-fremgangsmåten dårlig. Det burde i disse situasjonene undersøkes om det finnes andre bedre metoder for å estimere posisjonene til radarene. Foreløpig analyse er beskrevet i appendiks G.2.

Når ankomsttidene inneholder slengere, blir fordelingen for målestøy ikke-gaussisk. I oppgaven er denne fordelingen kun beskrevet numerisk. For å finne den teoretisk beste oppnåelige estimeringsnøyaktigheten i disse tilfeller, må CRLB beregnes for ikke-gaussiske fordelinger. Det kan tenkes at metoden beskrevet i [11] vil være egnet til dette formålet.

I oppgaven har det kun blitt benyttet simulerte data. Det vil alltid være forhold som ikke blir modellert i en simulator, og som påvirker resultatene. Det bør derfor undersøkes hvordan metoden fungerer på reelle data, og om det er noen forhold som gjør at radarer ikke lar seg geolokalisere. F.eks. ved at sensorene mister en eller flere hovedlobepasseringer fra radarene.

Den utviklede algoritmen har dårlig ytelse når det gjelder tidsbruk. Dersom algoritmen skal benyttes til analyser av reelle data, vil trolig datamengden bli mye større enn hva som var tilfellet for analysene som ble utført i denne rapporten. Algoritmen benytter mye tid på å bestemme initielle posisjoner ved den hierarkiske klyngeanalysen. Denne delen av algoritmen bør derfor byttes ut med et rektangulært grid med initielle posisjoner som ble benyttet i [2].

11 Konklusjon

Denne masteroppgaven bygger på den tidligere rapporten [2], hvor ulike algoritmer for å geolokalisere navigasjonsradarer vha. TDOA fra to fly ble analysert. I [2] ble det vist at TDOA fra syv radarer lot seg assosiere, og at det var mulig i finne de syv posisjonene blant en mengde mulige posisjoner. Derimot i [2] ble det ble ikke undersøkt hvilke begrensninger metoden hadde. Denne masteroppgaven har tatt for seg å finne sammenhenger mellom de viktigste forhold som avgjør hvor mange samtidige radarer som lar seg geolokalisere når det kun benyttes TDOA mellom to sensorplattformer i bevegelse til å assosiere TDOA-ene. Til å støtte oppunder analysene er det blitt utviklet en algoritme for å assosiere pulser fra flere samtidige radarer, for deretter å geolokalisere disse.

Oppgaven ser på to anvendelsesområder; geolokalisering av skip med navigasjonsradarer fra to små koordinerte UAS-er og fra to små tandem-satellitter. Målet med oppgaven er å assosiere pulser fra flere samtidige radarer slik at disse kan geolokaliseres, og ikke selve geolokaliseringen. Til å støtte oppunder analyser, er det laget en simulator til å generere ankomsttider til de to sensorplattformene for begge anvendelsesområdene. For enkelthetsskyld er det da benyttet flat jord for begge anvendelsesområdene, ettersom de samme modellene og algoritmene da kan benyttes. Dette blir en forenkling av begge scenarioer, men fører til enklere formler, mindre regnetid og mer tid til de egentlige problemstillingene.

I oppgaven er det utviklet og testet en metode for å assosiere pulser fra flere samtidige radarer og geolokalisere disse. Den utviklede metoden inneholder to deler; først en del hvor TDOA mellom pulser blir beregnet og hvor TDOA-er fra samme radar assosieres og midles, og andre del hvor midlede TDOA-er fra radarene i løpet av et helt måleopptak assosieres og benyttes til å geolokalisere radarene. Andre del bygger også på deler av metoden beskrevet i [2] ved at det beregnes en rekke initielle posisjoner som en estimator tar utgangspunkt i, men til forskjell fra [2] er estimeringsalgoritmen basert på en greedy-filosofi. Algoritmen fjerner da TDOA-er etter hvert som de blir benyttet til å estimere radarposisjonene.

Til å analysere hvor godt algoritmen fungerer, og identifisere typiske forhold som algoritmen har problemer med, er det benyttet Monte Carlo simuleringer på ulike datasett for begge anvendelsene. Simuleringene har vist at algoritmen stort sett gir riktige estimater av radarposisjonene, forutsatt at TDOA-ene ikke overlapper. Analysene har i disse tilfeller vist at estimatoren er forventningsrett og at estimeringsnøyaktigheten er i overensstemmelse med CRLB. De riktige estimatene viste seg å ofte benytte minst 5 ganger flere TDOA-er enn de gale estimatene. Ulempen ved algoritmevalget er når sensorene mottar TDOA-er fra flere radarer som overlapper. Det vil si at hyperblene fra en eller flere radar går gjennom eller nært en annen radar. I disse tilfellene gir algoritmen ofte et estimat som tilhører flere radarer, eller kun et estimat til den ene radarposisjonen. Det er vanskelig å si noe om hvor tett radarene kan ligge for at algoritmen skal estimere riktige estimater til hver av dem, ettersom det kommer an på hvor mange radarer som ligger i området og som blander seg inn med sine TDOA-er. Dersom det kun er to radarer vil en typisk minimumsavstand i UAS-scenarioet være et par km og i satellitt-scenarioet opp mot 50 km. I analysene som har blitt utført ser det ut til at minimum 30-50 % av TDOA-ene fra hver av radarene ikke bør overlappe for at algoritmen skal gi riktige estimater av hver av radarene.

Masteroppgaven har vist at det er tilstrekkelig å bare bruke TDOA for å assosiere pulser fra flere samtidige radarer, slik at disse kan geolokaliseres. Dette forutsetter at radarene ligger slik til at sensorene mottar TDOA-er fra radarene, som ikke assosieres til andre radarer. Dersom dette er tilfelle, er det ikke funnet noen begrensing i hvor mange samtidige radarer som kan geolokaliseres. Derimot hvis TDOA-ene overlapper og assosieres til feil radarer, gir algoritmen ofte et felles estimat for begge radarene eller at bare en av radarene estimeres. I disse tilfellene kan det trolig oppnås bedre resultater ved å benytte andre tilnærminger enn greedy-filosofien. I oppgaven er det kun identifisert et tilfelle hvor flere radarer ikke lar seg geolokalisere ved å bare bruke TDOA, nemlig når disse lå så tett at deres feilellipser overlapper og det vil da ikke være mulig å skille radarene ved å kun bruke TDOA.

Referanser

[1] E. O. Løvli, "Geolocalization with two airborne ESM-sensors," Norwegian university of science and technology, NTNU, 2005.

[2] E. O. Løvli and T. Smestad, "(U) Geolokalisering med tidsdifferanse mellom to fly - analyser og dataprogram (Begrenset)," Forsvarets forskningsinstitutt, 2006.

[3] T. Smestad, "Preliminary results of TDOA-geolocation by two satellites related to work of a "summer-student"," Forsvarets forskningsinstitutt, 2013.

[4] T. Sathyan, A. Sinha and T. Kirubarajan, "Passive geolocation and tracking of an unknown number of emitters," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 42, no. 2, pp. 740-750, 2006.

[5] O. Nickens and D. Musicki, "Emitter geolocation with two UAVs," IEEE, 2007.

[6] F. Fletcher, B. Ristic and D. Musicki, "Recursive estimation of emitter location using TDOA measurements from two UAVs," in *Information Fusion, 2007 10th International Conference on*, 2007.

[7] A. K. Bhattacharyya, Phased array antennas: Floquet analysis, synthesis, BFNs and active array systems, vol. 179, John Wiley \& Sons, 2006.

[8] C. A. Balanis, Antenna theory: analysis and design, John Wiley \& Sons, 2012.

[9] T. Smestad, "Konsept(er) for bruk av ESM i en AIS-satelitt for skipsovervåkning," Forsvarets forskningsinstitutt, 2013.

[10] C. Cheung, "Summer student 2013 Summary," Forsvarets forskningsinstitutt, 2013.

[11] M. Fowler, Binghamton University, 01 Mars 2006. [Online]. Available: http://www.ws.binghamton.edu/fowler/fowler%20personal%20page/EE522_files/EECE%20522%20Notes_15%20ML%20for%20TDOA-FDOA.pdf. [Accessed 15 Mai 2015].

[12] G. Høye, A. Smith, B. Oving, A. v. Kleef, T. Smestad, R. Olsen, R. Sundgot, M. Kloster and F. Gulbrandsen, "Deliverable 6.3.1: Operational Requirements and RF Landscaping," FFI, NLR, TNO, 2014.

[13] E. Malnes and P. S. T. Berg, "Metoder for sortering av pulstog (Deinterleaving)," 1998.

[14] N. Levanon, Radar Principles, John Wiley & Sons, Inc, 1988.

[15] O. Hallingstad, "Matematisk modellering av dynamiske systemer," Universitetssenteret på Kjeller, 2012.

[16] G. Høye, "Analyses of the geolocation accuracy that can be obtained from shipborne sensors by use of time difference of arrival (TDOA), scanphase, and angle of arrival (AOA) measurements," Forsvarets forskningsinstitutt, 2010.

[17] O. Hallingstad, "Måleoppdatering for Bayes, Fisher og VMK," Universitetssenteret på Kjeller, 2003.

[18] T. Smestad, ""Lynkurs"/kollokvium om "Kramer-Rao nedre grense" (CRLB)," Forsvarets forskningsinstitutt, 2003.

[19] O. Hallingstad, "Normalfordelinger, kovariansmatriser og ellipsoider," Universitetssenteret på Kjeller, 2005.

[20] T. Smestad, *MATLAB-program: Sammenheng mellom CEP og feilellipser*, Forsvarets forskningsinstitutt, 1999.

[21] S. Rogers, "Comments on "Computation of the circular error probability integral" by JT Gillis," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 2, pp. 553-555, 1993.

[22] MATLAB R2013a, *Introduction to Hierarchical Clustering*, MathWorks, 2013.

Appendiks

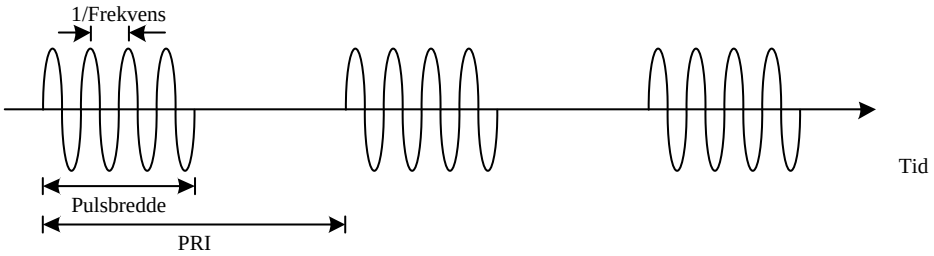
A Navigasjonsradarer

I følge Safety of Life at Sea (SOLAS) er alle kommersielle skip er påbudt å bruke en X-båndsradar. I tillegg må også alle fiskeskøyter i norsk farvann bruke en X-båndsradar. Større havgående fartøy benytter ofte en S-båndsradar i tillegg til X-båndsradarer. X-båndsradarene har en frekvens i området 9,3-9,5 GHz. Tabell A.1 inneholder typiske parametere for navigasjonsradarer i X-båndet.

Tabell A.1: Typiske parametere for navigasjonsradarer i X-båndet [12, p. 25].

Parameter	**Verdi**
Operasjonsavstand	≤ 120 nautisk mil (22 mil)
Antennelengde	1-3 m
Lobebredde (horisontalt)	1-6°
Lobebredde (vertikalt)	20-25°
Antennegain	29-35 dBi
Sendereffekt	10-50 kW
Frekvens	9,35-9,45 GHz
Pulsbredde	0,08-1,2 µs
Pulsrepetisjonsfrekvens (PRF)	500-3000 Hz
Rotasjonstid	1,9-2,8 s

Radarer har gjerne en variabel PRI for å eliminere blinde hastigheter og for å redusere effekten av jamming [13]. En radar varierer PRI-en ved to teknikker; PRI-stagger og PRI-jitter. Ved PRI-stagger endres PRI-en på en regelmessig i et fast mønster. F.eks. hopper PRI-en mellom faste nivåer i et fast mønster. PRI-stagger endres PRI-en med mer eller mindre tilfeldige variasjoner opp til 20-30 % av middelverdien til PRI-en. Figur A.1 viser et pulstog sendt fra en radar.



Figur A.1: Pulstog med tre pulser sendt fra en radar [14, p. 9].

B Notasjon og matematisk grunnlag

Notasjon og matematisk grunnlag er hentet fra [15].

B.1 Ramme

En ramme, eller et koordinatsystem, a i det affine rom benevnes $\mathcal{F}^a = \{O_a; \vec{a}_1, \vec{a}_2, \vec{a}_3\}$ og består av et origo O_a og basisvektorer $\vec{a}_i$.

B.2 Geometrisk vektor

Geometriske vektorer er piler i vektorrommet, og er uavhengig av valg av basisvektorer. En geometrisk vektor fra a til b betegnes $\vec{x}_{ab}$.

B.3 Algebraisk vektor

En algebraisk vektor er tallverdien av en geometrisk vektor projisert ned på et basisvektorsett. Algebraiske vektorer er derfor avhengig av valg av basisvektorer. Den geometriske vektoren $\vec{x}_{ab}$ projisert ned på basisvektorene i ramme $\mathcal{F}^a$ betegnes $\underline{x}_{ab}^a$.

B.4 Retningskosinmatrise (RKM)

Det er tre mulige tolkninger av RKM. I denne oppgaven benyttes det kun ortogonale retningskosinmatriser, og RKM betegnes derfor ved R_b^a.

Koordinattransformasjonsmatrise
R_b^a er en koordinattransformasjonsmatrise (KTM) som gir sammenhengen mellom de algebraiske vektorene $\underline{r}^b$ og $\underline{r}^a$:

$$\underline{r}^a = R_b^a \underline{r}^b \tag{B.1}$$

Stillingsmatrise
R_b^a er en stillingsmatrise som gir stillinga til ramme b i ramme a:

$$R_b^a = \left[\underline{b}_1^a, \underline{b}_2^a, \underline{b}_3^a\right] \tag{B.2}$$

Rotasjonsmatrise
$R_b^a = [\boldsymbol{R}_{ab}]^a = [\boldsymbol{R}_{ab}]^b$ er en rotasjonsmatrise som roterer en vektor i a-rammen til b-rammen:

$$\vec{r}_2 = \boldsymbol{R}_{ab}\vec{r}_1 \tag{B.3}$$

$$\underline{r}_2^a = R_b^a \underline{r}_1^a \tag{B.4}$$

$$\underline{r}_2^b = R_b^a \underline{r}_1^b \tag{B.5}$$

B.5 Eulerrepresentasjon av RKM

Antar at rammene $\mathcal{F}^a = \{O_a; \vec{a}_1, \vec{a}_2, \vec{a}_3\}$ og $\mathcal{F}^b = \{O_b; \vec{b}_1, \vec{b}_2, \vec{b}_3\}$ opprinnelig er sammenfallende. RKM fra ramme b til a kan da fås ved først å rotere ramme a en vinkel α_i om b_i-aksen, deretter en vinkel β_j den nye a_j-aksen og til slutt en vinkel γ_k om a_k-aksen:

$$R_b^a(\alpha_i, \beta_j, \gamma_k) = R_i(\alpha_i) R_j(\beta_j) R_k(\gamma_k), \tag{B.6}$$

hvor $R_i(\theta_i)$ er de elementære RKM gitt ved:

$$R_1(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{B.7}$$

$$R_2(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{B.8}$$

$$R_3(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.9}$$

For (ortogonale) RKM gjelder.

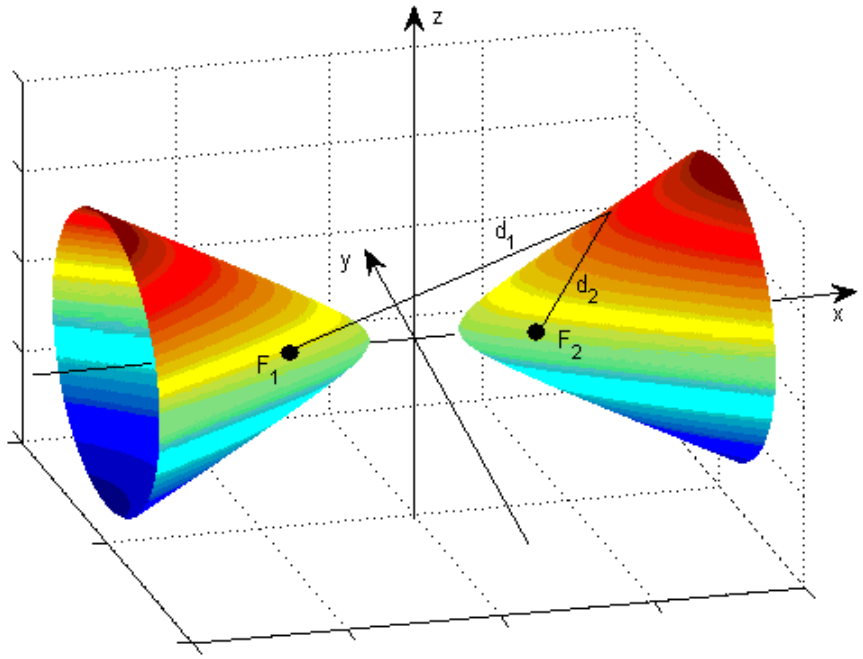
$$\big(R_i(\theta)\big)^{-1} = \big(R_i(\theta)\big)^T = R_i(-\theta) \tag{B.10}$$

Det betyr at KTM fra ramme b til ramme a kan skrives:

$$R_b^a = \big(R_a^b\big)^{-1} = \big(R_a^b\big)^T \tag{B.11}$$

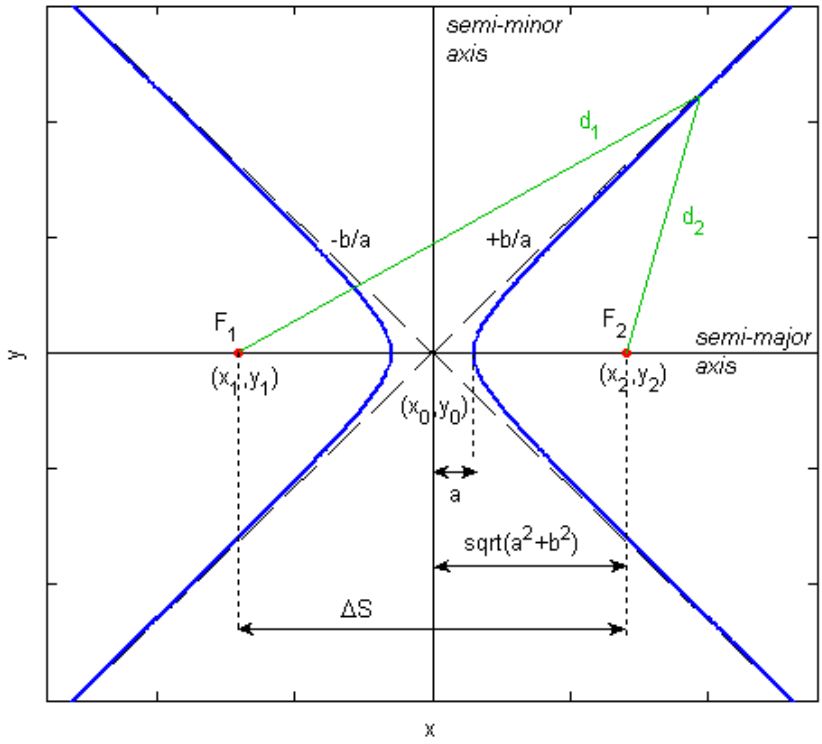
C Hyperboloider og hyperbler

Figur C.2 viser en hyperboloide og brennpunktene F_1 og F_2. En hyperboloide defineres ved at avstanden mellom et punkt og de to brennpunktene, hhv. d_1 og d_2, alltid er konstant. I likning (5.3) beregnes TDOA som tidsdifferansen fra en puls når den ene sensoren til den når den andre sensoren. Brennpunktene i figuren vil da være de to sensorposisjonene. Hyperboloidene inneholder mulige radarposisjoner som alle vil gi den samme TDOA. Merk at den ene hyperboloiden gjelder for positive TDOA, mens den andre gjelder for negative TDOA.



Figur C.2: Hyperboloide [16, p. 103].

I kapittel 5 ble det gjort den antakelse om at radaren lå på jordoverflaten. Likning (5.3) gir dermed løsninger som ligger på en hyperbel som er skjæringa mellom hyperboloidene og jordoverflaten. Figur C.3 viser to hyperbler som begge er skjæringa mellom hyperboloidene og et plan. I figuren har brennpunktene F_1 og F_2 lik høyde over planet med hyperblene. Hyperblene vil da speiles om de to aksene semi-minor og semi-major.

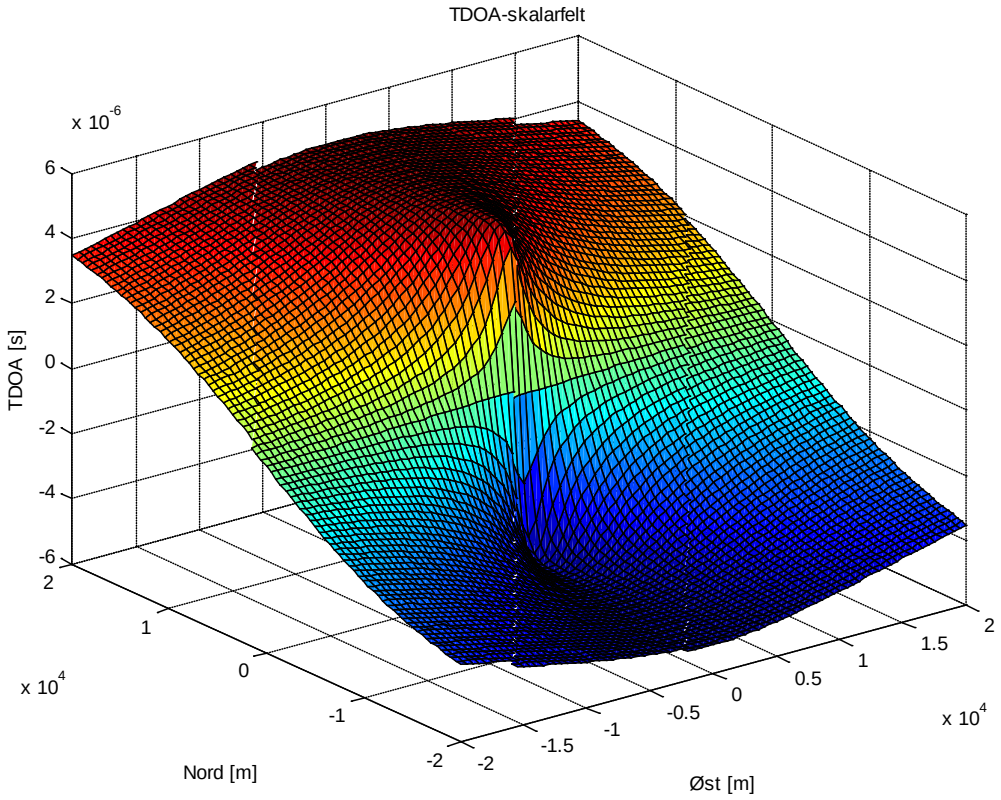


Figur C.3: Hyperbler [16, p. 100].

En hyperbel kan beregnes ved å beregne et skalarfelt for TDOA i ulike gridpunkter. MATLAB-funksjonen «contour» beregner deretter koordinatene til en hyperbel for en angitt TDOA. Skalarfeltet $h(\underline{x})$ i gridpunkt $\underline{x}$ beregnes ved

$$h(\underline{x}) = \frac{1}{c}\left(\left\|\underline{x} - \underline{p}_{S2}^{N}\right\| - \left\|\underline{x} - \underline{p}_{S1}^{N}\right\|\right), \tag{C.12}$$

hvor c er lyshastigheten, $\underline{p}_{S1}^{N}$ er posisjonen til sensor 1 og $\underline{p}_{S2}^{N}$ er posisjonen til sensor 2. Figur C.4 viser et typisk TDOA-skalarfelt. Hyperblene vil da være nivåkurver til skalarfeltet.



Figur C.4: TDOA-skalarfelt for ulike radarposisjoner.

D Estimeringsteori

D.1 Fishermodeller

En ulineær statisk Fishermodell er gitt ved [17]:

$$\underline{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_m \end{bmatrix} = \begin{bmatrix} h_1(\underline{x}) \\ \vdots \\ h_m(\underline{x}) \end{bmatrix} + \begin{bmatrix} w_1 \\ \vdots \\ w_M \end{bmatrix} = \underline{h}(\underline{x}) + \underline{w}, \qquad \underline{w} \sim p_w(\underline{w}), \tag{D.13}$$

hvor $\underline{x}$ er en fullstendig ukjent konstant vektor, og $\underline{w}$ er additiv støy med fordeling $p_w(\underline{w})$. I formel (D.13) er $\underline{z}$ en målevektor og $\underline{h}(\underline{x})$ en ulineær vektorfunksjon.

D.2 Estimat av tilstand i Fishermodeller

Gitt den ulineære statiske Fishermodellen fra formel (D.13). Estimatet av den ukjente tilstanden $\underline{x}$ kan da beregnes ved maksimum likelihood estimering :

$$\hat{\underline{x}} = \arg\max_{\underline{x}} p(\underline{z}; \underline{x}), \tag{D.14}$$

hvor $p(\underline{z}; \underline{x})$ er likelihoodfunksjonen. Dersom støyen er gaussisk med middelverdi $\underline{0}$ og kovariansmatrise R vil likelihoodfunksjonen være bestemt ved

$$p(\underline{z}; \underline{x}) = \frac{1}{\sqrt{(2\pi)^m |R|}} \exp\left(-\frac{1}{2} \left(\underline{z} - \underline{h}(\underline{x}) \right)^T R^{-1} \left(\underline{z} - \underline{h}(\underline{x}) \right) \right), \tag{D.15}$$

hvor m er dimensjonen til $\underline{z}$. Dersom målingene er ukorrelert og har lik varians, vil estimatet $\hat{\underline{x}}$ være den tilstanden $\underline{x}$ som minimaliserer kvadratavviket, altså

$$\hat{\underline{x}} = \arg\min_{\underline{x}} \left(\underline{z} - \underline{h}(\underline{x}) \right)^T \left(\underline{z} - \underline{h}(\underline{x}) \right). \tag{D.16}$$

Kovariansen til estimeringsfeilen blir da [18]

$$P = E\left\{ (\hat{\underline{x}} - \underline{x})(\hat{\underline{x}} - \underline{x})^T \right\}, \tag{D.17}$$

hvor $\underline{x}$ er den sanne tilstanden.

D.3 Cramér-Rao Lower Bound

Cramér-Rao Lower Bound (CRLB) gir den teoretiske nedre grensen for estimeringsnøyaktigheten i en forventningsrett estimator for Fishermodeller. I [18] utledes CRLB for Fishermodeller med additiv gaussisk støy, og følgende formler er hentet derfra. Den nedre grensen er den inverse av Fishers informasjonsmatrise

$$P_{est} \geq FIM^{-1}. \tag{D.18}$$

Fishers informasjonsmatrise sier noe om hvor mye informasjon som ligger i målingene i et system og bestemmes ved

$$FIM = -E\left\{\frac{\partial^2 \ln p(\underline{z}:\underline{x})}{\partial \underline{x} \partial \underline{x}^T}\right\}. \tag{D.19}$$

Dersom støyen er gaussisk vil kovariansen til estimeringsfeilen tilfredsstille

$$P_{est} \geq [D^T R^{-1} D]^{-1}, \tag{D.20}$$

hvor R er målestøyens kovariansmatrise og D beregnes ved

$$D = \frac{\partial \underline{z}}{\partial \underline{x}^T} = \left[\frac{\partial z_i}{\partial x_j}\right] = \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_m}{\partial x_1} & \cdots & \frac{\partial z_m}{\partial x_n} \end{bmatrix}. \tag{D.21}$$

D.4 Kovariansmatriser og feilellipser

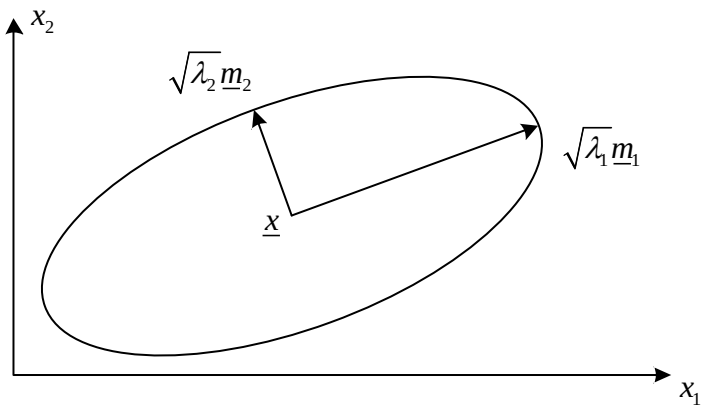
For gaussiske fordelinger er feilellipser «ekvi-sigma» ellipser beskrevet ved feilkovariansmatrisen [18]

$$p(\underline{\hat{x}}) = \frac{1}{\sqrt{(2\pi)^n |P|}} \exp\left(-\frac{1}{2}(\underline{\hat{x}} - \underline{x})^T P_x^{-1} (\underline{\hat{x}} - \underline{x})\right), \tag{D.22}$$

hvor $\underline{x}$ er den sanne tilstanden. En multidimensjonal feilellipsoide kan projiseres ned i en lavere dimensjon [19]. Figur D.5 viser en 2-dimensjonal feilellipse sentrert om den sanne tilstanden $\underline{x}$. I følge [19] kan ellipsen beskrives ved egenvektormatrisen til feilen mellom estimerte tilstanden og den sanne tilstanden

$$\underline{\hat{x}} - \underline{x} = M\underline{q}, \tag{D.23}$$

hvor M er den ortonormale egenvektormatrisen. Halvaksene til ellipsen bestemmes ved egenverdiene, $\lambda_i = \sigma_i^2$, mens orienteringen bestemmes ved egenvektorene, $\underline{m_i}$, til egenvektormatrisen.



Figur D.5: 2-dimensjonal feilellipse sentrert om den sanne tilstanden $\underline{x}$ [19].

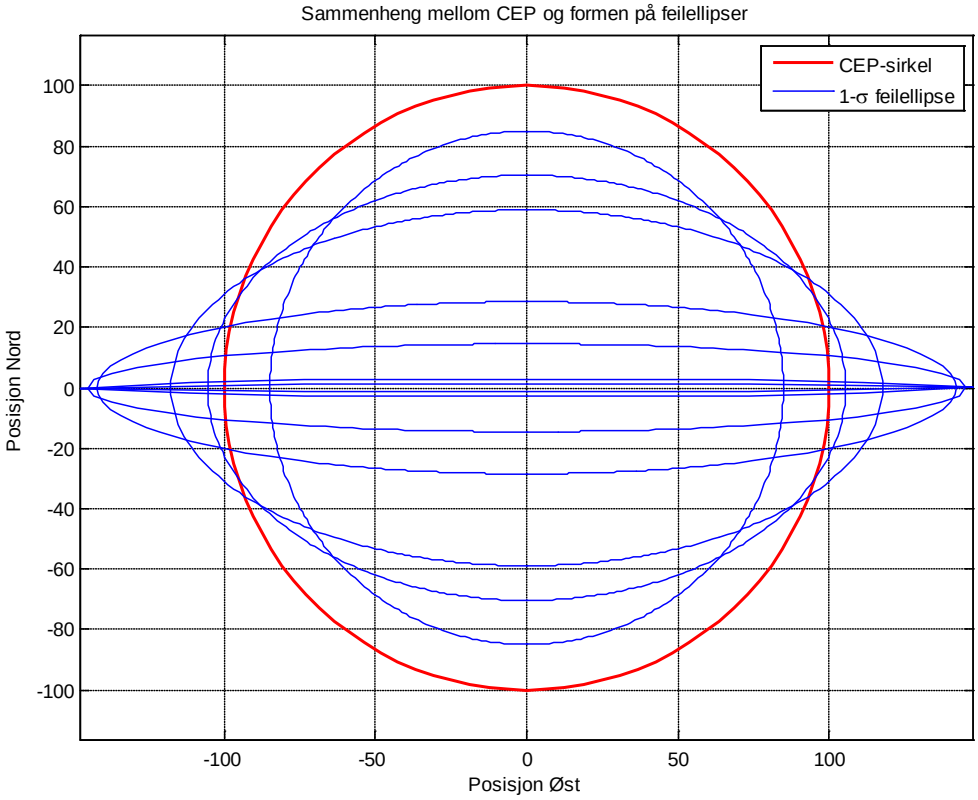
Tabell D.2 viser sannsynligheten for at en estimert tilstand ligger innenfor en 1σ, 2σ og 3σ feilellipsoide for et 1-, 2- og 3-dimensjonalt system.

Tabell D.2: Sannsynlighet innenfor l-σ ellipsoider [19]. n er dimensjonen til tilstanden $\underline{x}$ og l er antall σ.

n/l	1	2	3
1	0,683	0,955	0,997
2	0,394	0,865	0,989
3	0,200	0,739	0,971

D.5 Circular Error Probable

Circular Error Probable (CEP) er definert som en sirkel med en radius slik at det er 50 % sannsynlig at en 2-dimensjonal estimert tilstand ligger innenfor sirkelen. CEP-sirkelen gir ingen retningsinformasjon, men bevarer de statistiske egenskapene. CEP-brukes for at posisjonsusikkerheten skal kunne angis ved en skalar istedenfor en kovariansmatrise. Figur D.6 viser syv 1σ feilellipser som alle gir samme CEP.



Figur D.6: 1σ feilellipser som gir samme CEP-sirkel [20].

CEP-radiusen kan bestemmes numerisk ved [21]

$$p(r,\lambda_1,\lambda_2) = 1 - \frac{1}{N}\left(\frac{1}{2}f(\theta_0) + \sum_{k=1}^{N-1} f(\theta_k) + \frac{1}{1}f(\theta_N)\right), \qquad \theta_k = (k/N)\,\pi/2\,, k = 0,1,2,\ldots,N \tag{D.24}$$

hvor λ_1 og λ_2 er egenverdiene til kovariansmatrisen P_x, N er antall oppdelinger i intervallet $[0,\pi/2]$, og

$$f(\theta) = \exp\left(-\frac{r^2}{2(\lambda_1^2\cos^2\theta + \lambda_2^2\sin^2\theta)}\right). \tag{D.25}$$

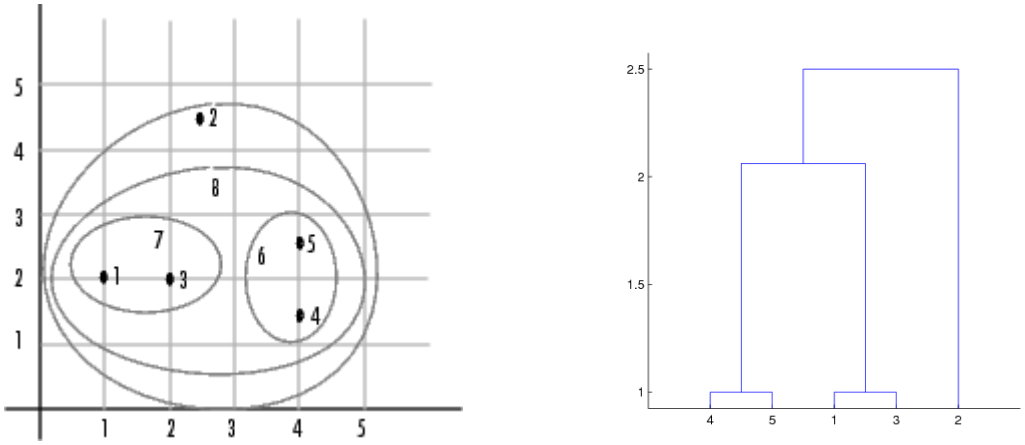
E Klyngeanalyse

Klyngeanalyse går ut på å fordele data inn i klynger, slik at en kriteriefunksjon minimaliseres. MATLAB har to hovedmetoder for klyngeanalyse:

1. Hierarkisk klyngeanalyse
2. K-means klyngeanalyse

Hierarkisk klyngeanalyse
En agglomerativ hierarkisk klyngeanalyse starter med like mange klynger som objekter i et datasett. Deretter beregner algoritmen avstanden mellom alle klyngene, f.eks. euklidsk avstand. De to nærmeste klyngene slås sammen, og antall klynger reduseres med en. Posisjonen til den nye klyngen kan beregnes på ulike måter, f.eks. som middelverdien av objektene i klyngen. Deretter beregnes avstanden mellom alle klynger på nytt, og de to nærmeste slås sammen. Slik fortsetter algoritmen inntil et stopp kriterie er oppfylt eller at det alle objektene er inneholdt i den samme klyngen.

Figur E.7 viser et datasett med seks objekter og tilhørende dendrogram. I figuren ligger klynge 1 og 3 nærmest, og disse slås sammen til klynge 7. Deretter slås klynge 4 og 5 sammen til klynge 6. Slik fortsetter algoritmen inntil et stopp kriterie er oppfylt.



Figur E.7: Venstre) Et datasett bestående av seks objekter og ni klynger [22]. Høyre) Dendrogram som viser avstanden mellom klyngene og hvilke klynger som slås sammen [22].

Fordelen med hierarkisk klyngeanalyse er at antall klynger ikke må spesifiseres, men at antallet bestemmes ut fra at et kriterie oppfylles. Algoritmen kan f.eks. avslutte dersom avstanden mellom middelverdiene til alle klyngene er over en grense. Ulempen er at nye objekter bare sammenliknes med eksisterende klynger og ikke objektene som utgjør klyngen. Det betyr at et objekt legges til den klynge som gir minst avstand mellom objektet og middelverdien til klyngen, og ikke den klyngen som f.eks. minimaliserer standardavviket.

K-means klyngeanalyse
I k-means klyngeanalyse spesifiseres antall klynger. Algoritmen fungerer ved at antall klynger på forhånd er spesifisert. Deretter flytter algoritmen objekter frem og tilbake mellom klyngene for å minimalisere et kriterie. Fordelen med k-means klyngeanalyse er at nye objekter sammenliknes med objektene i klyngen. Ulempen er at antall klynger må spesifiseres. Dersom antall klynger er ukjent, må antallet estimeres eller at ulike antall klynger testes ut.

F Oversikt over datamengde fra analyser

Tabell F.3 og Tabell F.4 inneholder en oversikt over data fra analysene utført i kapittel 7 og 9.

I Tabell F.3 inneholder rad 2 data fra avsnitt 7.1.2, rad 3 i data fra avsnitt 9.2.1, rad 4 data fra avsnitt 7.1.3 og rad 4data fra avsnitt 7.1.4.

I Tabell F.4 inneholder rad 2 data fra avsnitt 7.2.2, rad 3 i data fra avsnitt 9.3.1, rad 4 data fra avsnitt 7.2.3, rad 5 data fra avsnitt 7.2.4 og rad 6 data fra avsnitt 7.2.5.

Kolonne 1 inneholder antall radarer som er simulert, hvor (*) markerer at det er lagt på slengere på ankomsttidene. Kolonne 2 og 3 inneholder antall ankomsttider som er mottatt i simuleringen. Kolonne 4 inneholder antall beregnede TDOA, mens kolonne 5 inneholder antall TDOA som blir inkludert i klyngene i klyngeanalysen fra avsnitt 6.1. Kolonne 6 inneholder antall midlede TDOA. Kolonne 7 viser antall skjæringspunkter. Kolonne 8 viser antall initielle posisjoner og kolonne 9 viser antall estimerte posisjoner. I kolonne 10 inneholder gjennomsnittstiden til gjennomkjøringene av Monte Carlo simuleringen.

Tabell F.3: Oversikt over datamengde for UAS-scenario.

Antall radar	Antall TOA1	Antall TOA2	Antall TDOA	TDOA/ midlet TDOA	Antall midlet TDOA	Antall skjæ-ring	Antall init. pos.	Antall est. pos.	Tid [s]
1	2104	2203	1070	1063	35	509	9	2	1
1 (*)	2104	2203	1070	1015	36	539	23	4	2
6	10031	9976	5659	5379	254	16164	534	15	70
24	33438	33290	18872	17965	895	180402	2810	45	1456

I Tabell F.4 markerer (**) at de ti radarene er plassert tett i det opplyste området.

Tabell F.4: Oversikt over datamengde for satellitt-scenario.

Antall radar	Antall TOA1	Antall TOA2	Antall TDOA	TDOA/ midlet TDOA	Antall midlet TDOA	Antall skjæ-ring	Antall init. pos.	Antall est. pos.	Tid [s]
1	450	451	286	286	19	169	80	2	1
1 (*)	450	451	286	272	19	169	98	2	1
10	3957	4023	3317	2755	180	2782	2184	14	35
10 (**)	5582	5596	5467	4719	326	13053	10371	20	217
50	22794	22722	24874	18200	1396	68258	56054	123	14660

G Videre arbeid

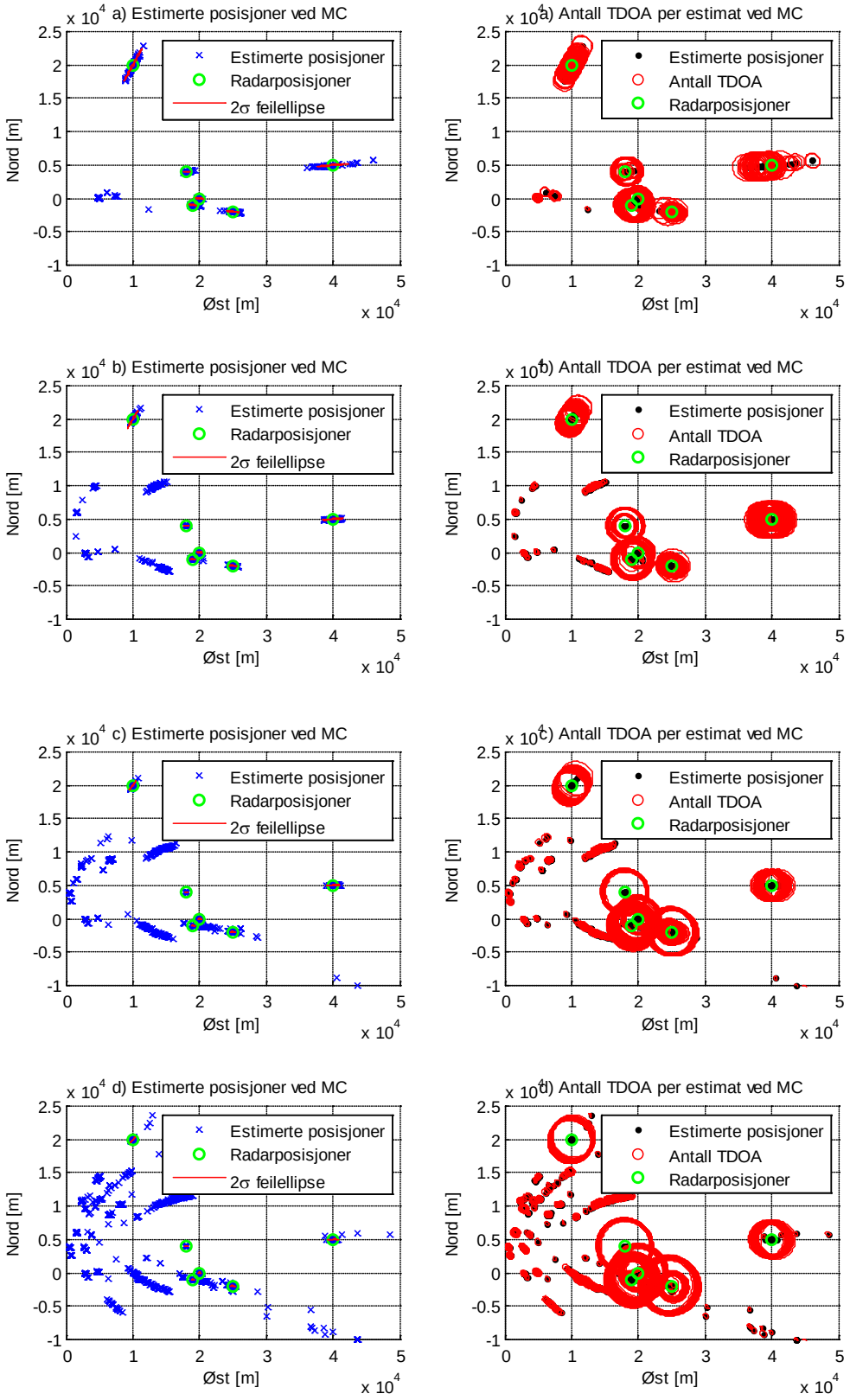
Avsnittene i kapittelet inneholder forslag til videre arbeid. Her presenteres kun noen foreløpige analyser som ikke er presentert i rapporten, ettersom analysene ikke er ferdigstilt.

G.1 Pulsassosiasjon og geolokalisering av radarer ved ulik antall måleintervall

I avsnitt 5.1.1 ble det nevnt at de to sensorene må motta pulser fra minimum tre ulike posisjoner for at radarene skal kunne geolokaliseres vha. TDOA. I oppgaven er det valgt at UAS-ene mottar pulser i måleintervaller. Det vil da være interessant å se hvor godt algoritmen klarer å geolokalisere radarene når UAS-ene mottar pulser fra ulikt antall måleintervaller. Figur G.8 viser åtte figurer, hvorav kolonnen til venstre viser estimerte posisjoner ved 100 gjennomkjøringer av Monte Carlo simulering kolonnen til høyre viser antall TDOA som er benyttet i hvert estimat. De fire radene viser det samme radaroppsettet, men hvor rad a) inneholder tre måleintervaller, rad b) fire målintervaller, rad c) fem måleintervaller og d) seks måleintervaller.

I de fire tilfellene, blir alle radarene estimert i hver gjennomkjøring, og det ser ut til at de riktige estimatene skiller seg fra de gale estimatene etter hvor mange TDOA som er benyttet i estimatene. Ved å øke antall måleintervall ser det ut til at det opprettes flere gale estimater, samtidig som de riktige estimatene skiller seg mer fra de gale ettersom det benyttes flere TDOA-er. Estimerings-nøyaktigheten ser også ut til å bli bedre når antall måleintervall øker.

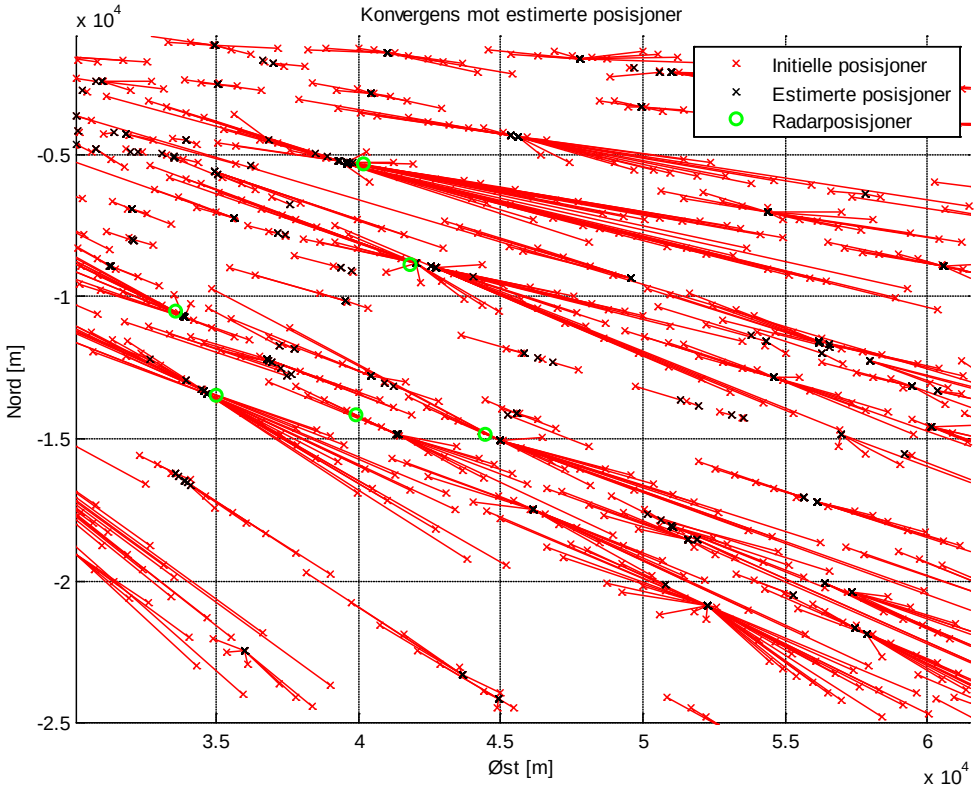
Det ser ut til at algoritmen geolokaliserer radarer som ligger langt fra hverandre, selv med få måleintervall. Men det bør undersøkes hvordan algoritmen fungerer på radarer som ligger tett ved få måleintervall, og hvilke begrensninger som gjelder.



Figur G.8: Estimerte posisjoner og antall TDOA per estimat ved Monter Carlo simulering.

G.2 Estimering av posisjoner med tilbakelegging av TDOA

I avsnitt 10.3 ble det diskutert hvorfor den utviklede algoritmen ikke klarte å estimere posisjoner som tydelig tilhørte radar 20, 21 og 22 fra analysen i avsnitt 7.1.4. Det ble der nevnt at radarposisjonene muligens kunne estimeres fra de initielle posisjonene uten å fjerne TDOA underveis. Figur G.9 viser estimerte posisjoner fra de initielle posisjonene når det ikke er fjernet noen TDOA. Mange av de initielle posisjonen konvergerer mot de samme estimatene, og de tre radarposisjonene får mange estimater. Samtidig gir mange initielle posisjoner også samme gale estimater. Mulige utfordringer vil være å finne ut hva som skiller disse riktige estimatene fra de gale estimatene, og hvordan TDOA-ene skal fordeles mellom de ulike estimatene (TDOA-ene har en radar de kommer fra, og bør dermed ikke benyttes til å estimere flere radarposisjoner).



Figur G.9: Sammenheng mellom initielle posisjoner og estimater når det ikke fjernes TDOA.

H Pseudokode

H.1 Simulering av ankomsttider i to UAS-er

Under følger pseudokode for simulering av én puls fra skip i i tidspunkt t_k.

Beregner posisjonsvektor til sensorramme 1 og 2
$\underline{p}_{S1}^{N}(t_k) = \underline{p}_{B1}^{N}(t_{k-1}) + R_{B1}^{N}\underline{v}_{B1}^{NB1}(t_k - t_{k-1}) + R_{B1}^{N}\underline{p}_{B1S1}^{B1}$, $\underline{p}_{B1}^{N}(t_0)$ er gitt.
$\underline{p}_{S2}^{N}(t_k) = \underline{p}_{B2}^{N}(t_{k-1}) + R_{B2}^{N}\underline{v}_{B2}^{NB2}(t_k - t_{k-1}) + R_{B2}^{N}\underline{p}_{B2S2}^{B2}$, $\underline{p}_{B2}^{N}(t_0)$ er gitt

Beregner KTM fra radarramme i til navigasjonsramme
$R_{Ri}^{N}(t_k) = R_{Ri}^{N}(t_k) = R_{Ri}^{N}(t_{k-1})R_y\big(\omega_i(t_k - t_{k-1})\big)$

Beregner propagasjonsvektorer i navigasjonsramma
$\underline{q}_{RiS1}^{N}(t_k) = \underline{p}_{S1}^{N}(t_k) - \underline{p}_{Ri}^{N}$
$\underline{q}_{RiS2}^{N}(t_k) = \underline{p}_{S2}^{N}(t_k) - \underline{p}_{Ri}^{N}$

Beregner propagasjonsvektorer i radarramme i
$\underline{q}_{RiS1}^{Ri}(t_k) = R_{N}^{Ri}(t_k)\underline{q}_{RiS1}^{N}(t_k)$
$\underline{q}_{RiS2}^{Ri}(t_k) = R_{N}^{Ri}(t_k)\underline{q}_{RiS2}^{N}(t_k)$

Beregner propagasjonsvektorer i sensorramme 1 og 2
$\underline{q}_{S1Ri}^{S1}(t_k) = R_{N}^{S1}\underline{q}_{RiS1}^{N}(t_k)$
$\underline{q}_{S2Ri}^{S2}(t_k) = R_{N}^{S2}\underline{q}_{RiS2}^{N}(t_k)$

Beregner vinkler mellom propagasjonsvektorer mot sensorramme j og radarramme i

$$\theta_{RiSj}(t_k) = \text{atan2}\left(\sqrt{\left(q_{RiSj}^{Ri}(1)\right)^2 + \left(q_{RiSj}^{Ri}(2)\right)^2}, q_{RiSj}^{Ri}(3)\right)$$

$$\phi_{RiSj}(t_k) = \text{atan2}\left(q_{RiSj}^{Ri}(2), q_{RiSj}^{Ri}(1)\right)$$

Beregner vinkler mellom propagasjonsvektorer mot radarramme i og sensorramme j

$$\theta_{SjRi}(t_k) = \text{atan2}\left(\sqrt{\left(q_{SjRi}^{Sj}(1)\right)^2 + \left(q_{SjRi}^{Sj}(2)\right)^2}, q_{SjRi}^{Sj}(3)\right)$$

$$\phi_{SjRi}(t_k) = \text{atan2}\left(q_{SjRi}^{Sj}(2), q_{SjRi}^{Sj}(1)\right)$$

Hvis $P_{S1,min} \leq P_{Ri}\left(\frac{\lambda}{4\pi}\right)^2\left(\left\|\underline{q}_{RiS1}^{N}(t_k)\right\|\right)^{-2} G_{Ri}\big(\theta_{RiS1}(t_k), \phi_{RiS1}(t_k)\big)G_{S1}\left(\theta_{S1Rj}(t_k), \phi_{S1Rj}(t_k)\right)$
  Beregne ankomsttid til sensor 1
  $TOA1 = t_k + \frac{1}{c}\left\|\underline{q}_{RiS1}^{N}(t_k)\right\|$

Hvis $P_{S1,min} \leq P_{Ri}\left(\frac{\lambda}{4\pi}\right)^2\left(\left\|\underline{q}_{RiS2}^{N}(t_k)\right\|\right)^{-2} G_{Ri}\big(\theta_{RiS2}(t_k), \phi_{RiS2}(t_k)\big)G_{S2}\left(\theta_{S2Rj}(t_k), \phi_{S2Rj}(t_k)\right)$
  Beregne ankomsttid til sensor 2
  $TOA2 = t_k + \frac{1}{c}\left\|\underline{q}_{RiS2}^{N}(t_k)\right\|$

H.2 Simulering av ankomsttider i to satellitter

Under følger pseudokode for simulering av én puls fra skip i i tidspunkt t_k.

Beregner posisjonsvektor til sensorramme 1 og 2
$\underline{p}_{S1}^{N}(t_k) = \underline{p}_{B1}^{N}(t_{k-1}) + R_{B1}^{N}\underline{v}_{B1}^{NB1}(t_k - t_{k-1}) + R_{B1}^{N}\underline{p}_{B1S1}^{B1}$, $\underline{p}_{B1}^{N}(t_0)$ er gitt.
$\underline{p}_{S2}^{N}(t_k) = \underline{p}_{B2}^{N}(t_{k-1}) + R_{B2}^{N}\underline{v}_{B2}^{NB2}(t_k - t_{k-1}) + R_{B2}^{N}\underline{p}_{B2S2}^{B2}$, $\underline{p}_{B2}^{N}(t_0)$ er gitt

Beregner KTM fra radarramme i til navigasjonsramme
$R_{Ri}^{N}(t_k) = R_{Ri}^{N}(t_k) = R_{Ri}^{N}(t_{k-1})R_y\big(\omega_i(t_k - t_{k-1})\big)$

Beregner propagasjonsvektorer i navigasjonsrammen
$\underline{q}_{RiS1}^{N}(t_k) = \underline{p}_{S1}^{N}(t_k) - \underline{p}_{Ri}^{N}$
$\underline{q}_{RiS2}^{N}(t_k) = \underline{p}_{S2}^{N}(t_k) - \underline{p}_{Ri}^{N}$

Beregner propagasjonstid
$t_{sat1} = \frac{1}{c}\left\|\underline{q}_{RiS1}^{N}(t_k)\right\|$
$t_{sat2} = \frac{1}{c}\left\|\underline{q}_{RiS2}^{N}(t_k)\right\|$

Flytter satellittene
$\underline{p}_{S1}^{N}(t_k') = \underline{p}_{S1}^{N}(t_k) + R_{B1}^{N}\underline{v}_{B1}^{NB1}t_{sat1}$
$\underline{p}_{S2}^{N}(t_k') = \underline{p}_{S2}^{N}(t_k) + R_{B2}^{N}\underline{v}_{B2}^{NB2}t_{sat2}$

Beregner propagasjonsvektorer i navigasjonsrammen
$\underline{q}_{RiS1}^{N}(t_k') = \underline{p}_{S1}^{N}(t_k') - \underline{p}_{Ri}^{N}$
$\underline{q}_{RiS2}^{N}(t_k') = \underline{p}_{S2}^{N}(t_k') - \underline{p}_{Ri}^{N}$

Beregner propagasjonsvektorer i radarramme i
$\underline{q}_{RiS1}^{Ri}(t_k') = R_{N}^{Ri}(t_k)\underline{q}_{RiS1}^{N}(t_k')$
$\underline{q}_{RiS2}^{Ri}(t_k') = R_{N}^{Ri}(t_k)\underline{q}_{RiS2}^{N}(t_k')$

Beregner propagasjonsvektorer i sensorramme 1 og 2
$\underline{q}_{S1Ri}^{S1}(t_k') = R_{N}^{S1}\underline{q}_{RiS1}^{N}(t_k')$
$\underline{q}_{S2Ri}^{S2}(t_k') = R_{N}^{S2}\underline{q}_{RiS2}^{N}(t_k')$

Beregner vinkler mellom propagasjonsvektorer mot sensorramme j og radarramme i
$\phi_{RiSj}(t_k') = \text{atan2}\left(q_{RiS1}^{Ri}(1), q_{RiS1}^{Ri}(3)\right)$

Beregner vinkler mellom propagasjonsvektorer mot radarramme i og sensorramme j
$$\theta_{SjRi}(t_k') = \text{atan2}\left(\sqrt{\left(q_{SjRi}^{Sj}(1)\right)^2 + \left(q_{SjRi}^{Sj}(2)\right)^2}, q_{SjRi}^{Sj}(3)\right)$$

Hvis $|\phi_{RiS1}(t_k')| \leq \frac{\theta_{Ri}}{2}$ og $|\theta_{S1Ri}(t_k')| \leq \frac{\theta_{S1}}{2}$
  Beregne ankomsttid til sensor 1
  $TOA1 = t_k + \frac{1}{c}\left\|\underline{q}_{RiS1}^{N}(t_k')\right\|$

Hvis $|\phi_{RiS2}(t'_k)| \leq \frac{\theta_{Ri}}{2}$ og $|\theta_{S2Ri}(t'_k)| \leq \frac{\theta_{S2}}{2}$

Beregne ankomsttid til sensor 1

$TOA2 = t_k + \frac{1}{c}\left\|\underline{q}^N_{RiS2}(t'_k)\right\|$

I MATLAB-koder

I MATLAB-koden betegnes den algebraiske vektoren $\underline{x}_{ab}^{a}$ ved `x_ab_a`, og KTM R_b^a ved `R_a_b`.

I.1 Pulsassosiasjon og geolokaliserings-algoritme

Hovedprogram:

- pulsassosiasjon_og_geolokalisering.m

Funksjoner:

- beregn_TDOA.m
- estimator.m
- h.m
- hyperbel.m
- hyperbel_intersect.m

Hovedprogram – pulsassosiasjon_og_geolokalisering.m

```
% pulsassosiasjon_og_geolokalisering.m
%
% Initialiseringer

clc         % Fjerner alt i command window
close all   % Lukker alle vinduer
clear all   % Sletter alle variable
format long % Long fixed decimal format
warning off % Skrur av eventuelle advarsler

global c
c = 299792458;  % Lyshastighet [m/s]

% Endepunkter for simulering og grid for UAS-scenario
x_lim = [ -25000 25000 ];            % Endepunkter i retning nord [m]
y_lim = [ 0 65000 ];                 % Endepunkter i retnin øst [m]
x_step = 500;                        % Oppløsning i retning nord [m]
y_step = 500;                        % Oppløsning i retning øst [m]

% Endepunkter for simulering og grid for satellitt-scenario
x_lim = [ -1000e3 1000e3 ];          % Endepunkter i retning nord [m]
y_lim = [ 0 3000e3 ];                % Endepunkter i retnin øst [m]
x_step = 100e3;                      % Oppløsning i retning nord [m]
y_step = 100e3;                      % Oppløsning i retning øst [m]

x_grid = x_lim(1):x_step:x_lim(2);   % Grid i retning nord [m]
y_grid = y_lim(1):y_step:y_lim(2);   % Grid i retning nord [m]

% Støy for målestøy uten slengere
TOA1_std  = 50e-9;                   % Standardavvik til TOA1
TOA2_std  = 50e-9;                   % Standardavvik til TOA2
TDOA_var  = TOA1_std^2 + TOA2_std^2; % Variansen til TDOA
TDOA_std  = sqrt(TDOA_var);          % Standardavvik til TDOA
TDOAm_var = TDOA_var/10;             % Varians til 10 midlede TDOA
TDOAm_std = sqrt(TDOAm_var);         % Standardavvik til 10 midlede TDOA

% Støy for målestøy med slengere
slenger = false;                     % True: Slengere / False: Uten slengere
TOA1_N_slenger = 0.1;                % Prosentvis andel slengere
TOA2_N_slenger = 0.1;                % Prosentvis andel slengere
TOA1_slenger_std = 200e-9;           % Standardavvik til slengere på TOA1
TOA2_slenger_std = 200e-9;           % Standardavvik til slengere på TOA2
```

```
% Datasett
theta_sat = deg2rad(10);              % Lobebredde til satellitter
sensor1 = load( '???.mat' );          % Data fra sensor 1
sensor2 = load( '???.mat' );          % Data fra sensor 2
load( '???.mat' );                    % Sann informasjon om radarer

type = 'uas';                         % 'uas' hvis UAS
type = 'sat';                         % 'sat' hvis satellitt

%% Monte carlo simulering
N_MC = 1;                             % Antall Monte Carlo gjennomkjøringer
for i_mc = 1:N_MC

  clear('data');
  data.N_intervall = length( sensor1.intervall );

  % Velger seed-nr lik i_mc
  rng(i_mc);



%% Legger på støy på målingene
  TOA1_slenger_indeks = [];
  TOA2_slenger_indeks = [];

  % Løper gjennom alle måleintervall og legger på støy
  for i = 1:data.N_intervall

    N1 = length(sensor1.intervall(i).TOAj);
    N2 = length(sensor2.intervall(i).TOAj);

    % Legger bare til støy hvis antall målinger er større enn 0
    if N1 > 0
      % Trekker additivt støy på TOA 1
      data.intervall(i).TOA1 = sensor1.intervall(i).TOAj + TOA1_std*randn(1,N1);

      % Indekser til TOA 1
      TOA1_slenger_indeks = [ TOA1_slenger_indeks, [ i*ones(1,N1); 1:N1 ] ];
    else
      data.intervall(i).TOA1 = [];
    end

    if N2 > 0
      % Trekker additivt støy på TOA 2
      data.intervall(i).TOA2 = sensor2.intervall(i).TOAj + TOA2_std*randn(1,N2);

      % Indekser til TOA 2
      TOA2_slenger_indeks = [ TOA2_slenger_indeks, [ i*ones(1,N2); 1:N2 ] ];
    else
      data.intervall(i).TOA2 = [];
    end

    clear('N1','N2');

  end

  % Legger til slengere hvis slenger == true
  if slenger

    % Antall ankomsttider i de to sensorene
    N_TOA1 = length(TOA1_slenger_indeks);
    N_TOA2 = length(TOA2_slenger_indeks);

    % Finner indeks til X% av tilfeldig trukne indekser
    TOA1_slenger_indeks = TOA1_slenger_indeks( :, ( rand(1,N_TOA1) <= TOA1_N_slenger ) );
    TOA2_slenger_indeks = TOA2_slenger_indeks( :, ( rand(1,N_TOA2) <= TOA2_N_slenger ) );

    % Legger til slengere på de trukne indeksene
    for i = 1:data.N_intervall

      % Indeks til ankomsttider med slenger i TOA 1
      data.intervall(i).TOA1_slenger_indeks = ...
        TOA1_slenger_indeks( 2, TOA1_slenger_indeks(1,:) == i );

      % Antall slengere i TOA 1
      data.intervall(i).TOA1_N_slenger = ...
        length( data.intervall(i).TOA1_slenger_indeks );
```

```
    % Indeks til ankomsttider med slenger i TOA 2
    data.intervall(i).TOA2_slenger_indeks = ...
      TOA2_slenger_indeks( 2, TOA2_slenger_indeks(1,:) == i );

    % Antall slengere i TOA 2
    data.intervall(i).TOA2_N_slenger = ...
      length( data.intervall(i).TOA2_slenger_indeks );

    % Legger til slengere på TOA 1
    data.intervall(i).TOA1( data.intervall(i).TOA1_slenger_indeks ) = ...
      sensor1.intervall(i).TOAj( data.intervall(i).TOA1_slenger_indeks ) + ...
      TOA1_slenger_std*randn(1,data.intervall(i).TOA1_N_slenger);

    % Legger til slengere på TOA 2
    data.intervall(i).TOA2( data.intervall(i).TOA2_slenger_indeks ) = ...
      sensor2.intervall(i).TOAj( data.intervall(i).TOA2_slenger_indeks ) + ...
      TOA2_slenger_std*randn(1,data.intervall(i).TOA2_N_slenger);
  end
end

%% Beregner TDOA

% Løper gjennom alle måleintervall
for i = 1:data.N_intervall

  % Deler opp i segmenter på 100 ms og beregner TDOA
  t = sensor1.intervall(i).t_intervall(1):0.1:sensor1.intervall(i).t_intervall(2);
  data.intervall(i).N_segment = length(t)-1;

  for j = 1:data.intervall(i).N_segment

    % Finner pulsnummer som tilhører segmentet
    i_toa1 = find( t(j) <= data.intervall(i).TOA1 & data.intervall(i).TOA1 < t(j+1) );
    i_toa2 = find( t(j) <= data.intervall(i).TOA2 & data.intervall(i).TOA2 < t(j+1) );

    % Beregner TDOA og finner indeks til pulser det kan beregnes TDOA fra
    [ data.intervall(i).segment(j).TDOA, i1, i2 ] = ...
      beregn_TDOA( data.intervall(i).TOA1(i_toa1), data.intervall(i).TOA2(i_toa2),...
      sensor1.intervall(i).d );

    % Finner data som inngår i TDOA-ene
    data.intervall(i).segment(j).TOA1 = data.intervall(i).TOA1(i_toa1(i1));
    data.intervall(i).segment(j).TOA2 = data.intervall(i).TOA2(i_toa2(i2));
    data.intervall(i).segment(j).puls_nr1 = i_toa1(i1);
    data.intervall(i).segment(j).puls_nr2 = i_toa2(i2);
    data.intervall(i).segment(j).p_S1_N = sensor1.intervall(i).p_Sj_N(:,i_toa1(i1));
    data.intervall(i).segment(j).p_S2_N = sensor2.intervall(i).p_Sj_N(:,i_toa2(i2));

    % Antall TDOA
    data.intervall(i).segment(j).N_TDOA = length( data.intervall(i).segment(j).TDOA );
  end
  clear('t','i1','i2','i_toa1','i_toa2')
end

%% Midler TDOA
% Reduserer antall TDOA-er og reduserer spredningen.

min_TDOA_seg = 4;    % Minimum antall TDOA i hvert segment
min_TDOA_klynge = 4; % Minimum antall TDOA i hver klynge
a_TDOA = 3;          % Distansemål (antall sigma) i klyngeanalysen

% Løper gjennom alle måleintervall
for i = 1:data.N_intervall

  % Tellevariabel for TDOA-grupper innad i måleintervallet
  k_teller = 0;

  % Løper gjennom alle segmenter innad i måleintervallet
  for j = 1:data.intervall(i).N_segment;

    % Antall tdoa-er >= min_TDOA_seg
    if data.intervall(i).segment(j).N_TDOA >= min_TDOA_seg

      % Klyngeanalyse
      %   Criterion:  Kriterie for å danne en ny klynge er distanse
      %   Cutoff:     Oppretter klynge når avstand er større enn a_TDOA*TDOA_std
      %   Distance:   Distansemålet er Euklidsk-avstand
```

```
      %   Linkage:    Avstanden mellom to klynger beregnes
      %               fra centroid (middelverdien) til klynga
      %   Savememory: For å spare minne (fungerer bare
      %               for euklidsk avstand og nærmeste nabo)
      klynge = clusterdata( data.intervall(i).segment(j).TDOA', 'criterion', ...
        'distance', 'cutoff', a_TDOA*TDOA_std, 'distance', 'euclidean', ...
        inkage', 'centroid', 'savememory', 'on' );

      % Antall ulike klynger
      K = length( unique( klynge ) );

      % Deler segmentene opp i K klynger
      for k = 1:K

        % Antall tdoa-er i klynga >= min_TDOA_klynge TDOA-er
        if length( find( klynge == k ) ) >= min_TDOA_klynge

          % Klynge nummer k_teller i måleintervallet
          k_teller = k_teller + 1;

          % TDOA i klynga
          data.intervall(i).klynge(k_teller).TDOA = ...
            data.intervall(i).segment(j).TDOA( klynge == k );

          % Antall TDOA i klynga
          data.intervall(i).klynge(k_teller).N_TDOA = ...
            length( data.intervall(i).klynge(k_teller).TDOA );

          % Legger over data som inngår i klynga
          data.intervall(i).klynge(k_teller).puls_nr1 = ...
            data.intervall(i).segment(j).puls_nr1( klynge == k );
          data.intervall(i).klynge(k_teller).puls_nr2 = ...
            data.intervall(i).segment(j).puls_nr2( klynge == k );

          data.intervall(i).klynge(k_teller).TOA1 = ...
            data.intervall(i).segment(j).TOA1( :, klynge == k );
          data.intervall(i).klynge(k_teller).TOA2 = ...
            data.intervall(i).segment(j).TOA2( :, klynge == k );

          data.intervall(i).klynge(k_teller).p_S1_N  = ...
            data.intervall(i).segment(j).p_S1_N( :, klynge == k );
          data.intervall(i).klynge(k_teller).p_S2_N  = ...
            data.intervall(i).segment(j).p_S2_N( :, klynge == k );

          % Middelverdi og standardavvik til TDOA i klynga
          data.intervall(i).klynge(k_teller).TDOAm = ...
            mean( data.intervall(i).klynge(k_teller).TDOA, 2 );
          data.intervall(i).klynge(k_teller).TDOAs = ...
            std ( data.intervall(i).klynge(k_teller).TDOA, [], 2 );

          % Middelverdier til sensorposisjonene i klynga
          data.intervall(i).klynge(k_teller).pm_S1_N = ...
            mean( data.intervall(i).klynge(k_teller).p_S1_N, 2 );
          data.intervall(i).klynge(k_teller).pm_S2_N = ...
            mean( data.intervall(i).klynge(k_teller).p_S2_N, 2 );

        end
      end
    end
  end

  data.intervall(i).N_klynge = k_teller;
  clear('klynge','k_teller');
end


%% Finner skjæringspunkter mellom hyperbler fra ulike måleintervall
% Antall skjæringspunkter bestemmes ved:
% N = N1*(N2+N3+N4+N5+...+Nn)+N2*(N3+N4+N5+...+Nn)+N3*(N4+N5+...+Nn)+...
%     +N4*(N5+...+Nn)+N5*(+...+Nn)

% Tellevariabel for antall skjæringspunkter
k_teller = 0;

% Finner mulige kombinasjoner av hyperbler

% Går gjennom alle måleintervall
```

```
for i = 1:data.N_intervall

  % Går gjennom alle klynger i måleintervall nr. i
  for j = 1:data.intervall(i).N_klynge

    % Går gjennom alle måleintervall etter måleintervall nr. i
    for k = i+1:data.N_intervall

      % Går gjennom alle klynger i måleintervall nr. k
      for l = 1:length(data.intervall(k).klynge);

        k_teller = k_teller + 1;
        data.intersect(k_teller).ik = [ [ i; j ], [ k; l ] ];

      end
    end
  end
end

data.N_intersect = k_teller;
clear('k_teller');

% Finner skjæringspunkter mellom gjennværende hyperbler
I = data.N_intersect;
flagg_intersect = false(1,I);
x_intersect = zeros(3,I);
for i = 1:I

  i1 = data.intersect(i).ik(1,1);
  k1 = data.intersect(i).ik(2,1);
  i2 = data.intersect(i).ik(1,2);
  k2 = data.intersect(i).ik(2,2);

  TDOAm =   [ data.intervall(i1).klynge(k1).TDOAm,   ...
              data.intervall(i2).klynge(k2).TDOAm   ];
  pm_S1_N = [ data.intervall(i1).klynge(k1).pm_S1_N, ...
              data.intervall(i2).klynge(k2).pm_S1_N ];
  pm_S2_N = [ data.intervall(i1).klynge(k1).pm_S2_N, ...
              data.intervall(i2).klynge(k2).pm_S2_N ];

  [ data.intersect(i).x_intersect, flagg_intersect(i) ] = ...
    hyperbel_intersect( TDOAm, pm_S1_N, pm_S2_N, x_grid, y_grid );
  x_intersect(:,i) = data.intersect(i).x_intersect;

  % Hvis det prosesseres data for satellitt, testes  det om skjæringspunktet ligger
  % innenfor loben til begge satellittene i begge måleintervallene.

  if strcmp(type,'sat')

    % Første måleintervall

    % Vektor fra skjæringspunkt til satellitt 1 i N
    q_xS1_N = pm_S1_N(:,1) - data.intersect(i).x_intersect;

     % Vektor fra skjæringspunkt til satellitt 2 i N
    q_xS2_N = pm_S2_N(:,1) - data.intersect(i).x_intersect;

    R_N_S1 = sensor1.intervall(i1).R_Sj_N'; % RKM fra N til S1
    R_N_S2 = sensor2.intervall(i1).R_Sj_N'; % RKM fra N til S2

    q_S1x_S1 = R_N_S1*(-q_xS1_N); % Vektor fra satellitt 1 til skjæringspunkt i S1
    q_S2x_S2 = R_N_S2*(-q_xS2_N); % Vektor fra satellitt 2 til skjæringspunkt i S2

    % Vinkel mellom pekeretning til hovedlobe på satellitt 1 og vektor mot skjæringspkt
    phi(1) = abs( atan2( norm( q_S1x_S1(1:2) ), q_S1x_S1(3) ) );

    % Vinkel mellom pekeretning til hovedlobe på satellitt 2 og vektor mot skjæringspkt
    phi(2) = abs( atan2( norm( q_S2x_S2(1:2) ), q_S2x_S2(3) ) );

    % Andre måleintervall

    % Vektor fra skjæringspunkt til satellitt 1 i N
    q_xS1_N = pm_S1_N(:,2) - data.intersect(i).x_intersect;

    % Vektor fra skjæringspunkt til satellitt 2 i N
    q_xS2_N = pm_S2_N(:,2) - data.intersect(i).x_intersect;
```

```
    R_N_S1 = sensor1.intervall(i2).R_Sj_N'; % RKM fra N til S1
    R_N_S2 = sensor2.intervall(i2).R_Sj_N'; % RKM fra N til S2

    q_S1x_S1 = R_N_S1*(-q_xS1_N); % Vektor fra satellitt 1 til skjæringspunkt i S1
    q_S2x_S2 = R_N_S2*(-q_xS2_N); % Vektor fra satellitt 2 til skjæringspunkt i S2

    % Vinkel mellom pekeretning til hovedlobe på satellitt 1 og vektor mot skjæringspkt
    phi(3) = abs( atan2( norm( q_S1x_S1(1:2) ), q_S1x_S1(3) ) );

    % Vinkel mellom pekeretning til hovedlobe på satellitt 2 og vektor mot skjæringspkt
    phi(4) = abs( atan2( norm( q_S2x_S2(1:2) ), q_S2x_S2(3) ) );

    % Tester om skjæringspunktet ligger innenfor hovedlobene til begge
    % satellittene i begge måleintervall. Videre må flagget allerede
    % være satt fra hyperbel_intersect.
    if flagg_intersect(i) == 1 && all( phi <= theta_sat/2 )
      flagg_intersect(i) = 1;
    else
      flagg_intersect(i) = 0;
    end
  elseif strcmp(type,'uas')
    if flagg_intersect(i) == 1
      flagg_intersect(i) = 1;
    else
      flagg_intersect(i) = 0;
    end
  end

end

clear('i1','i2','k1','k2','q_xS1_N','q_xS2_N','R_N_S1','R_N_S2','q_S1x_S1',...
  'q_S2x_S2','phi')

% Fjerner skjæringspunkter med "ikke-satt" flagg, dvs. at det ikke er
% funnet et skjæringspunkt, eller at skjæringspunktet ligger utenfor
% hovedlobene til satellittene i begge måleintervallene
data.intersect   = data.intersect( flagg_intersect );
data.N_intersect = length( data.intersect );
x_intersect      = x_intersect( :, flagg_intersect );

%% Finner grupper (klynger) av skjæringspunkter
maks_avstand = 500;

% Klyngeanalyse
%   Criterion:  Kriterie for å danne en ny klynge er distanse
%   Cutoff:     Lager en ny klynge når et element er lengre unna enn maks_avstand
%   Distance:   Distansemålet er Euklidsk-avstand
%   Linkage:    Avstanden mellom to klynger beregnes fra centroid
%               (middelverdien) til klynga
%   Savememory: For å spare minne (fungerer bare for euklidsk avstand og
%               nærmeste nabo)
gruppe = clusterdata( x_intersect(1:2,:)', 'criterion', 'distance', 'cutoff', ...
  maks_avstand, 'distance', 'euclidean', 'linkage', 'centroid', 'savememory', 'on' );

data.gruppe = [];

% Antall unike gruppe
I = length( unique( gruppe ) );
for i = 1:I

  intersect_indeks = find( gruppe == i )';
  data.gruppe(i).N_intersect = length(intersect_indeks);

  data.gruppe(i).x_intersect  = x_intersect( :, intersect_indeks );
  data.gruppe(i).xm_intersect = mean( data.gruppe(i).x_intersect, 2 );
  data.gruppe(i).xs_intersect = std( data.gruppe(i).x_intersect, [], 2 );

end

data.N_gruppe = length(data.gruppe);
clear('intersect_indeks')

%% Estimerer posisjon

% Liste over alle midlede TDOA og tilhørende posisjoner
TDOAm   = [];
pm_S1_N = [];
```

```
pm_S2_N = [];
ik      = [];

for i = 1:data.N_intervall

  for j = 1:data.intervall(i).N_klynge

    TDOAm   = [ TDOAm,   data.intervall(i).klynge(j).TDOAm   ];
    pm_S1_N = [ pm_S1_N, data.intervall(i).klynge(j).pm_S1_N ];
    pm_S2_N = [ pm_S2_N, data.intervall(i).klynge(j).pm_S2_N ];
    ik      = [ ik, [ i; j ] ];

  end
end

% Setter opp en liste med mulige grupper
gruppe_pre = data.gruppe;

% Antall sigma som benyttes i algoritmen
a_std = 2;
% Tellevariabel
i = 1;

stopp_kriterie = false;
while ~stopp_kriterie

  % Kriterieverdi
  xi = zeros(1,data.N_gruppe);

  % Tilordner TDOA-er som ligger innenfor +/- a_std sigma til
  % initielle posisjoner
  for j = 1:length(gruppe_pre)

    % Finner sann TDOA (dersom radaren lå i den initielle posisjonen)
    TDOAm_sann = h( gruppe_pre(j).xm_intersect, pm_S1_N, pm_S2_N );

    % Finner indeksen til TDOA som ligger innenfor +/- a*TDOAm_std
    indeks = abs( TDOAm - TDOAm_sann ) <= a_std*TDOAm_std;

    % Tilordner data til gruppa
    gruppe_pre(j).TDOAm      = TDOAm( indeks );
    gruppe_pre(j).TDOAm_sann = TDOAm_sann( indeks );
    gruppe_pre(j).ik         = ik( :, indeks );
    gruppe_pre(j).pm_S1_N    = pm_S1_N( :, indeks );
    gruppe_pre(j).pm_S2_N    = pm_S2_N( :, indeks );
    gruppe_pre(j).N_TDOAm(:,i)    = length( gruppe_pre(j).TDOAm );

    % Antall måleintervall representert i gruppa
    gruppe_pre(j).N_intervall(:,i)  = length( unique( gruppe_pre(j).ik(1,:) ) );

    % Antall TDOA fra hvert måleintervall
    [ ~, ~, ic ] = unique( gruppe_pre(j).ik(1,:) );
    gruppe_pre(j).N_klynge = hist( ic, gruppe_pre(j).N_intervall(:,i) );

    % Beregner RMS
    gruppe_pre(j).RMS(:,i) = ...
      norm( gruppe_pre(j).TDOAm - gruppe_pre(j).TDOAm_sann )/...
      sqrt( gruppe_pre(j).N_TDOAm(:,i)  );

    % ML-kritere
    min_N_TDOAm = 2;
    min_N_intervall = 2;

    % Hvis antall TDOA >= min_N_TDOAm og antall måleintervall >= min_N_intervall
    if gruppe_pre(j).N_TDOAm(:,i) > min_N_TDOAm & ...
      gruppe_pre(j).N_intervall(:,i) > min_N_intervall

      % Kriterie
      gruppe_pre(j).xi(:,i) = gruppe_pre(j).N_TDOAm(:,i);
    else
      gruppe_pre(j).xi(:,i) = 0;
    end

    % Kriterieverdi
    xi(j) = gruppe_pre(j).xi(:,i);

  end
```

```
    % Hvis ikke alle kriterieverdier er 0
    if sum(xi) ~= 0

      % Plukker ut den posisjonen med størst verdi på kriteriefunksjonen xi
      [ ~, max_indeks ] = max( xi );

      % Estimat av posisjon
      [ gruppe_pre(max_indeks).x_est, gruppe_pre(max_indeks).J_val, ...
        gruppe_pre(max_indeks).flagg, indeks_brukt, gruppe_pre(max_indeks).TDOAm, ...
        gruppe_pre(max_indeks).TDOAm_sann, gruppe_pre(max_indeks).pm_S1_N, ...
        gruppe_pre(max_indeks).pm_S2_N, gruppe_pre(max_indeks).RMS ] = ...
        estimator( gruppe_pre(max_indeks).xm_intersect(1:2), TDOAm, pm_S1_N, ...
        pm_S2_N, a_std, TDOAm_std );

      % Hvis søkealgoritme fant et minimum
      if gruppe_pre(max_indeks).flagg > 0
        % Overfører estimat til ferdig behandlet gruppe
        gruppe_post(i) = gruppe_pre(max_indeks);

        % Fjerner brukte data (greedy-filosofi)
        %   Dersom det ønskes å beholde brukte data, komenter ut de følgende 4 linjene.
        TDOAm( indeks_brukt ) = [];
        pm_S1_N( :, indeks_brukt ) = [];
        pm_S2_N( :, indeks_brukt ) = [];
        ik( :, indeks_brukt ) = [];

        i = i + 1;
      end

      % Sletter brukt data
      gruppe_pre(max_indeks) = [];

    else
      stopp_kriterie = true;
    end

  end

  mc(i_mc).gruppe_post = gruppe_post;
  x_est(:,i_mc) = gruppe_post(1).x_est(:,end);

end
```

Funksjon – beregn_TDOA.m

```
function [ TDOA, iTOA1, iTOA2 ] = beregn_TDOA( TOA1, TOA2, d )
% [ TDOA, iTOA1, iTOA2 ] = beregn_TDOA( TOA1, TOA2 )
%
% Beregner TDOA mellom ankomsttider i sensor 1 (TOA1) og
% ankomsttider i sensor 2 (TOA2). TDOA er definert som TOA2 - TOA1 slik at
% hvis en puls treffer sensor 1 før sensor 2, blir TDOA > 0.
%
% Merk at det kan være ulikt antall TOA1 og TOA2.
%
% TDOA-ene må være mindre eller lik, i absouluttverdi, enn tiden
% det tar for lyset å propagere i den rette linjen mellom sensorene.
%
% Input:
%  - TOA1: Ankomsttider til sensor 1.
%  - TOA2: Ankomsttider til sensor 2.
%  - d:    Avstand [m] mellom sensor 1 og sensor 2.
%
% Output:
%  - TDOA:  Tidsdifferanse mellom TOA1 og TOA2 ( TDOA = TOA2 - TOA1 ).
%  - iTOA1: Indeksene til TOA1 i TDOA.
%  - iTOA2: Indeksene til TOA2 i TDOA.
%           TDOA(1) = TOA2( iTOA2 ) - TOA1( iTOA1 )
%

global c

% Maksimal TDOA (tiden det tar for lyset å propagere i den rette linje mellom sensorene)
TDOA_maks = d/c;

% Setter opp TOA1 og TOA2 på meshgrid-form
[ TOA1, TOA2 ] = meshgrid( TOA1(:), TOA2(:) );
```

```
% TDOA-matrise.
TDOA = TOA2 - TOA1; % TDOA(i,:) = TOA2(i)-TOA1(:). TDOA(:,j) = TOA2(:)-TOA1(j)

% Finner hvilke TOA1 og TOA2 som gir TDOA
[ iTOA2, iTOA1 ] = find( abs(TDOA) <= TDOA_maks );

% Finner hvilke TDOA-er som i absoluttverdi er mindre eller lik TODA_maks.
TDOA = TDOA( abs(TDOA) <= TDOA_maks )';

end
```

Funksjon – estimator.m

```
function [ x_est, J_val, flagg, indeks, TDOA, TDOA_sann, p_S1_N, p_S2_N, RMS ] = ...
    estimator( x0, TDOA_liste, p_S1_N_liste, p_S2_N_liste, antall_sigma, TDOA_std )
% [ x_est, J_val, flagg, indeks, TDOA, TDOA_sann, p_S1_N, p_S2_N, RMS ] = ...
%   estimator( x0, TDOA_liste, p_S1_N_liste, p_S2_N_liste, antall_sigma, TDOA_std )
%
% Estimerer posisjon ved å minimalisere kvadratavviket. Estimatoren velger
% ut TDOA som ligger innenfor +- a simga fra sann TDOA dersom radaren hadde
% stått i den initelle posisjonen. Deretter estimeres en posisjon. Denne
% posisjonen fungerer som en initiell posisjon, og estimatoren begynner på
% nytt. Estimatoren stopper når det ikke er noen endring i estimert
% posisjon, eller etter et maks antall iterasjoner.

% Input:
%  - x0           : Initiell posisjon (i 2D: [nord;øst] )
%  - TDOA_liste   : TDOA som kan brukes
%  - p_S1_N_liste : Tilhørende sensorposisjon 1
%  - p_S2_N_liste : Tilhørende sensorposisjon 1
%  - antall_sigma : Antall std som skal inkluderes
%  - TDOA_std     : Standardavvik
%
% Output:
%  - x_est     : Estimert posisjon (i 2D)
%  - J_val     : Kriterium til vmk
%  - flagg     : Flagg fra fminseacrh
%  - indeks    : Antall TDOA
%  - TDOA      : Benyttede TDOA
%  - TDOA_sann : Sann TDOA
%  - p_S1_N    : Benyttede sensorposisjon 1
%  - p_S2_N    : Benyttede sensorposisjon 2
%  - RMS       : RMS til benyttede TDOA
%

x_est  = x0;
J_val  = 0;
flagg  = 1;
indeks = [];
TDOA   = [];
TDOA_sann = [];
p_S1_N = [];
p_S2_N = [];
RMS    = [];

% Minimum perturbasjon
options = optimset( 'TolX', 1e-6 );

% Nedre grense for endring av estimat
tol = 1e-6;

i = 2;
stopp_kriterie = false;
while ~stopp_kriterie

  % Finner sann TDOA
  TDOA_sann = h( x_est(:,i-1), p_S1_N_liste, p_S2_N_liste );

  % Finner indeksen til TDOA som ligger innenfor +/- a*TDOA_std
  indeks = abs( TDOA_liste - TDOA_sann ) <= antall_sigma*TDOA_std;

  TDOA      = TDOA_liste( indeks );
  TDOA_sann = TDOA_sann( indeks );
  p_S1_N    = p_S1_N_liste( :, indeks );
  p_S2_N    = p_S2_N_liste( :, indeks );
  N         = length(TDOA);
```

```
  RMS = norm( TDOA - TDOA_sann )/sqrt( N );

  % Estimerer posisjon ved minstekvadraters metode. Estimatet blir da den x som
  % minimaliserer kvadratavviket
  J = @(x)kvadratavvik( x, TDOA, p_S1_N, p_S2_N );
  [ x_est(:,i), J_val(:,i), flagg(:,i) ] = fminsearch( J, x_est(:,i-1), options );

  if abs( x_est(:,i) - x_est(:,i-1) ) < tol
    stopp_kriterie = true;
  elseif i > 10
    stopp_kriterie = true;
  else
    i = i + 1;
  end

end

end


% xi = kvadratavvik( x, z, p_S1_N, p_S2_N )
%
% Kvadratavvik

function xi = kvadratavvik( x, z, p_S1_N, p_S2_N )

% Forventet måling, h(x), (lik som funksjonen h, men i 2d)
hx = h( x, p_S1_N, p_S2_N );

% Kvadratavvik
xi = norm(z-hx)^2;

end
```

Funksjon – h.m

```
function TDOA = h( p_Ri_N, p_S1_N, p_S2_N )
% TDOA = h( p_Ri_N, p_S1_N, p_S2_N )
%
% Beregner tidsdifferanse basert på radarens posisjon og to sensorers
% posisjoner. Tidsdifferansen er definert slik at dersom pulsen treffer
% sensor 1 først, blir TDOA > 0.
%
% Input:
%  - p_Ri_N: Posisjon til radaren.    Dimmensjon: [o,1] (o = 2 eller 3 ).
%  - p_S1_N: Posisjonen til sensor 1. Dimmensjon: [m,n].
%  - p_S2_N: Posisjonen til sensor 2. Dimmensjon: [m,n].
%
% Output:
%  - TDOA:   Tidsdifferanse. Dimmensjon: [1,n].
%

global c

if size(p_Ri_N) < 3
  p_Ri_N = [ p_Ri_N; 0 ];
end

J = size(p_S1_N,2);
TDOA = zeros(1,J);

for j=1:J

  % Avstand mellom sensor 1 og radar
  d1 = norm( p_Ri_N - p_S1_N(:,j) );

  % Avstand mellom sensor 2 og radar
  d2 = norm( p_Ri_N - p_S2_N(:,j) );

  % Tidsdifferansen
  TDOA(j) = ( d2 - d1 )/c;
end
end
```

Funksjon – hyperbel_intersect.m

```
function [ x, flag ] = hyperbel_intersect( TDOA, p_S1_N, p_S2_N, x_grid, y_grid, z_grid )
% [ x, flag ] = hyperbel_intersect( TDOA, p_S1_N, p_S2_N, x_grid, y_grid, z_grid )
%
% Bestemmer skjæringspunktet mellom to hyperbler, hvis det finnes.
%
% Input:
%  - TDOA:   2 TDOA-er
%  - p_S1_N: 2 posisjoner til sensor 1
%  - p_S2_N: 2 posisjoner til sensor 2
%  - x_grid: Grid i x-retning
%  - y_grid: Grid i y-retning
%  - z_grid: Grid i z-retning (bare lik 0)
%
% Output:
%  - x:    Koordinat til skjæringspunkt
%  - flag: Satt hvis skjæringspunkt er funnet
%

if nargin < 6
  z_grid = 0;
end

flag = false;
x = nan*zeros(3,1);

% Beregner TDOA i grid for posisjon 1
x_hyp1 = hyperbel( TDOA(1), p_S1_N(:,1), p_S2_N(:,1), x_grid, y_grid, z_grid );

% Beregner TDOA i grid for posisjon 2
x_hyp2 = hyperbel( TDOA(2), p_S1_N(:,2), p_S2_N(:,2), x_grid, y_grid, z_grid );

if isempty(x_hyp1) || isempty(x_hyp2)
  return;
end

[ y_int, i1, i2 ] = intersect( x_hyp1(2,:), x_hyp2(2,:) );

% Plukker ut x-verdiene med like y-verdier
x_int1 = x_hyp1(1,i1);
x_int2 = x_hyp2(1,i2);
x_diff = x_int2 - x_int1;

% Finner punktet før og etter x_diff skjærer y-aksen
I = length(x_diff);
for i = 1:I-1

  if sign(x_diff(i)) ~= sign(x_diff(i+1))

    flag = true;

    y0 = y_int(i);
    y1 = y_int(i+1);
    x0 = x_int1(i);
    x1 = x_int1(i+1);
    xd0 = x_diff(i);
    xd1 = x_diff(i+1);

    x(2) = -(y1-y0)/(xd1-xd0)*xd0 + y0;
    x(1) = (x1-x0)/(y1-y0)*(x(2)-y0) + x0;
    x(3) = 0;

  end
end

end
```

Funksjon – hyperbel.m

```
function x = hyperbel( TDOA, p_S1_N, p_S2_N, x_grid, y_grid, z_grid )
% x = hyperbel( TDOA, p_S1_N, p_S2_N, x_grid, y_grid, z_grid )
%
% Bestemmer skjæringspunktet mellom to hyperbler, hvis det finnes.
%
% Input:
```

```
%  - TDOA:   TDOA
%  - p_S1_N: Posisjon til sensor 1
%  - p_S2_N: Posisjon til sensor 2
%  - x_grid: Grid i x-retning
%  - y_grid: Grid i y-retning
%  - z_grid: Grid i z-retning (bare lik 0)
%
% Output:
%  - x:  Koordinater til hyperbelen
%

if nargin < 6
  z_grid = 0;
end

[ X, Y, Z ] = meshgrid( x_grid, y_grid, z_grid );

% Beregner TDOA i grid
TDOA_grid = h_meshgrid( X, Y, Z, p_S1_N, p_S2_N );
C = contourc( x_grid, y_grid, TDOA_grid, [ TDOA, TDOA ] );

if ~isempty(C)
  x = C(:,2:end);
else
  x = [];
end

end
```

I.2 CRLB for UAS-scenario

Hovedprogram:

- CRLB_uas.m

Funksjoner:

- CRLB.m
- CEP_radius.m
- finn_CEP.m
- ellipse.m
- sirkel.m

Hovedprogram – CRLB_uas.m

```
% CRLB_uas.m
%
% Beregner CRLB og CEP for alle skipsposisjoner. Antar at radarene har en rotasjonstid
% på 2.2s.

load( '???.mat' );            % Sann informasjon om radarer

TDOA_var  = 2*(50e-9)^2;      % Varians for TDOA
TDOAm_var = TDOA_var/10;      % Varians for 10 midlede TDOA

p0_S1_N = [  0;    0; -500 ]; % Startposisjon for UAS 1
p0_S2_N = [ -1500; 0; -500 ]; % Startposisjon for UAS 2

v_S1_NS1  = [ 20; 0; 0 ];     % Hastighetsvektor [m/s]
v_S2_NS2  = [ 20; 0; 0 ];     % Hastighetsvektor [m/s]

T = 510;                      % Sluttidspunkt
radar_t = 0:2.2:T;            % Tidspunkt for hovedlobepasseringer

% Finner tidspunktene som UAS-ene mottar hovedlobepasseringer
% (fordelt over seks intervaller)
```

```
t = [ radar_t(   0 <= radar_t & radar_t <= 10 ), ...
      radar_t( 100 <= radar_t & radar_t <= 110 ), ...
      radar_t( 200 <= radar_t & radar_t <= 210 ), ...
      radar_t( 300 <= radar_t & radar_t <= 310 ), ...
      radar_t( 400 <= radar_t & radar_t <= 410 ), ...
      radar_t( 500 <= radar_t & radar_t <= 510 ) ];

for i = 1:length(skip);

  % Antall hovedlobepasseringer
  skip(i).N_lobe = length(t);

  % Posisjon til UAS-er i tidspunkt for hovedlobepasseringene
  p_S1_N = p0_S1_N*ones(1,skip(i).N_lobe) + v_B1_NB1*t;
  p_S2_N = p0_S2_N*ones(1,skip(i).N_lobe) + v_B2_NB2*t;

  % Finn kovariansmatrise, cep-radius, feilellipser og cep-sirkel
  [ skip(i).Px, skip(i).Rcep, skip(i).x_s1, skip(i).x_s2, skip(i).x_cep ] = ...
    CRLB( p_S1_N, p_S2_N, skip(i).p_Ri_N, TDOAm_var );

end
```

Funksjon – CRLB.m

```
function [ Px, Rcep, x_s1, x_s2, x_cep ] = CRLB( p_S1_N, p_S2_N, p_x_N, TDOA_var )
% [ Px, Rcep, x_s1, x_s2, x_cep ] = CRLB( p_S1_N, p_S2_N, p_x_N, TDOA_var )
%
% Beregner CRLB
%
% Input:
%  - p_S1_N:   Posisjon til sensor 1 [3,N]
%  - p_S2_N:   Posisjon til sensor 2 [3,N]
%  - p_x_N:    Posisjon til radar
%  - TDOA_var: Varians til TDOA
%
% Output:
%  - Px:    Kovariansmatrise [2,2]
%  - Rcep:  Cep-radius
%  - x_s1:  Koordinater for 1s feilellipse [x-vektor;y-vektor]
%  - x_s2:  Koordinater for 2s feilellipse [x-vektor;y-vektor]
%  - x_cep: Koordinater for cep-sirkel [x-vektor;y-vektor]
%

% Perturbasjoner
dx  = 0.001;
dx1 = [ dx; 0; 0 ];
dx2 = [ 0; dx; 0 ];

% CRLB beregnes kun hvis antall målinger er større enn 1 og hvis sensorposisjonene
% eksisterer
I = size(p_B1_N,2);
if I > 1 & ~isnan(p_B1_N)

  % Kovarkansmatrise for måling
  Pw = TDOA_var*eye(I);

  h0   = h( p_x_N,       p_S1_N, p_S2_N )';
  hdx1 = h( p_x_N + dx1, p_S1_N, p_S2_N )';
  hdx2 = h( p_x_N + dx2, p_S1_N, p_S2_N )';

  % Jacobimatrisen
  D = [ hdx1-h0 hdx2-h0 ]./dx;

  % CRLB
  Px   = inv( D'/Pw*D );

  % CEP-radius
  Rcep = CEP_radius(Px);
  x_s1  = ellipse( p_x_N, Px, 1 );
  x_s2  = ellipse ( p_x_N, Px, 2 );
  x_cep = sirkel( p_x_N(1:2), Rcep );

else
  Px   = nan;
  Rcep = nan;
```

```
  x_s1  = nan(2,1);
  x_s2  = nan(2,1);
  x_cep = nan(2,1);
end

end
```

Funksjon – CEP_radius.m

```
function R = CEP_radius( Px )
% R = CEP_radius( Px, P )
%
% Beregner CEP-radius fra en 2x2 kovariansmatrise
%
% Input:
%  - Px: 2x2 kovariansmatrise

% Output:
%  - R: Radius
%

P = @(R)finn_CEP(R,Px);
R = fminsearch(P,1);

end
```

Funksjon – finn_CEP.m

```
function J = finn_CEP(R,Px)
% J = finn_CEP(R,Px)
%
% Beregner CEP numerisk.
%
% Se: Rogers, SR. Comments on "Computation of the circular error probability integral"
% by JT Gillis. IEEE Transactions on Aerospace and Electronic Systems. 1993,
% Vol. 29, 2, pp. 553-555.
%
% Input:
%  - R:  Radius det søkes etter
%  - Px: 2x2 kovariansmatrise
%
% Output:
%  - J: Kriterie
%

lambda = eig(Px);
a = sqrt(lambda(1));
b = sqrt(lambda(2));

N = 10;
k = 0:1:N;

theta = k/N*pi/2;
f = exp( -R^2./( 2*( a^2*cos(theta).^2 + b^2*sin(theta).^2 ) ) );
P = 1 - ( f(1)/2 + sum(f(2:end-1)) + f(end)/2 )/N;

J = (P-0.5)^2;

end
```

Funksjon – ellipse.m

```
function x = ellipse( x_senter, P, a )
% x = ellipse( x_senter, a, P )
%
% Returnerer en [2xN] vektor med x-array og y-array for plotting av en
% a*sigma ellipse bestemt av kovariansmatrisen P [2x2].
%
% Input
%  - x_senter: Posisjonsvektor til senter av ellipsoida [2x1].
%  - a:        a*sigma-ellipsoide.
%  - P:        Kovariansmatrise [2x2].
%
```

```
% Output:
%  - x:        En matrise hvor øverste rad inneholder x-koordianter og
%              nederste rad inneholder y-koordinater [2xN].
%

% Kovariansmatrise
Pinv = inv(P);

% Setter opp koordinater
dv = pi/1000;
cv = 1./tan( [ dv : dv : pi ] );

x(1,:) = [ 0, sqrt( 1./( Pinv(1,1) + 2*Pinv(1,2)*cv + Pinv(2,2)*cv.^2 ) ), 0 ];
x(2,:) = [ sqrt( 1/Pinv(2,2) )*sign(x(1,2)), ...
           cv.*x(1,2:end-1), ...
           -sqrt( 1/Pinv(2,2) )*sign(x(1,2)) ];
x = [ x, -x(:,2:end-1) ];

x(1,:) = x_senter(1) + a*x(1,:);
x(2,:) = x_senter(2) + a*x(2,:);

end
```

Funksjon – sirkel.m

```
function x = sirkel( x0, r )
% x = sirkel( x0, r, n )
%
% Returnerer en sirkel med senter i x0 og radius r

v = linspace( 0, 2*pi, 2000 );

x(1,:) = x0(1) + r*cos(v);
x(2,:) = x0(2) + r*sin(v);

end
```

I.3 CRLB for satellitt-scenario

Hovedprogram:

- CRLB_sat.m

Funksjoner:

- kulejord2flatjord..m
- opplyst_omrade.m

Funksjoner (fra appendiks I.2):

- CRLB.m
- CEP_radius.m
- finn_CEP.m
- ellipse.m
- sirkel.m

Hovedprogram – CRLB_sat.m

```
% CRLB_sat
%
% Beregner CRLB og CEP for alle skipsposisjoner. Antar her at radarene har
% en rotasjonstid på 2.2s.
```

```
load( '???.mat' );            % Sann informasjon om radarer

% Generelle satellittparametere
[ h_sat, v_sat, p_xc_N, R_S1_N, R_S2_N, beta ] = kulejord2flatjord;

% Initielle satellitt-posisjoner
p0_S1_N = [  15e3; 0; -h_sat ];
p0_S2_N = [ -15e3; 0; -h_sat ];

% Finner opplyst område
[ lobe_sat1, lobe_sat2, lobe_felles, X, Y, Z ] = ...
  opplyst_omrade( p0_S1_N, p0_S2_N, R_S1_N, R_S2_N, x_grid, y_grid );

% Belyst område
lobe_felles = contourc( y_grid, x_grid, lobe_felles', [ theta_sat/2, theta_sat/2 ] );

% Fjnerner overflødig informasjon
lobe_felles = lobe_felles(:,2:end);

% Kun positiv halvsirkel
ind = find( lobe_felles(2,:) >= 0 );
lobe_felles = lobe_felles(1:2,ind);

% Sorterer data i stigende rekkefølge
lobe_felles = sortrows(lobe_felles')';

% Fjerner duplikate data
lobe_felles = unique( lobe_felles', 'rows' )';

for i = 1:length(skip);

  % Flytter satellittene slik at de står symmetrisk om skipsposisjonen i
  % retning nord
  p0_S1_N = [  d_sat/2 + skip(i).p_Ri_N(1); 0; -h_sat ];
  p0_S2_N = [ -d_sat/2 + skip(i).p_Ri_N(1); 0; -h_sat ];

  % Bredde på felles lobe, i retning nord, ved skipsposisjon
  b = 2*interp1( lobe_felles(1,:), lobe_felles(2,:), skip(i).p_Ri_N(2) );

  skip(i).t_tot = b/v_sat;                              % Total belyst tid
  rot_tid = 2.2;                                        % Rotasjonstid til radar
  t = 0:rot_tid:floor(skip(i).t_tot/rot_tid)*rot_tid;   % Tidsakse
  t = t-t(end)/2;
  skip(i).N_lobe = length(t);                           % Antall hovedlober

  % Beregner satellittposisjon under hovedlobepasseringene
  p_S1_N = p0_S1_N*ones(1,skip(i).N_lobe) + [v_sat;0;0]*t;
  p_S2_N = p0_S2_N*ones(1,skip(i).N_lobe) + [v_sat;0;0]*t;

  % Finn kovariansmatrise, cep-radius, feilellipser og cep-sirkel
  [ skip(i).Px, skip(i).Rcep, skip(i).x_s1, skip(i).x_s2, skip(i).x_cep ] = ...
    CRLB( p_S1_N, p_S2_N, skip(i).p_Ri_N, TDOAm_var );

end
```

Funksjon – kulejord2flatjord.m

```
function [ h_sat, v_sat, p_xc_N, R_S1_N, R_S2_N, beta ] = kulejord2flatjord
% [ h_sat, v_sat, p_xc_N ] = kulejord2flatjord
%
% Beregner satellittparametere for flat jord

% Jordparametere
r = 6371e3; % Jordradius

% Satellittparametere
h_sat = 600e3;  % Satellitt-banehøyde
d     = 30e3;   % Avstand mellom satellittene
v_sat = 7.5e3;  % Banehastighet;

% Hastighet nede på jordoverflaten
v_sat = r/(r+h_sat)*v_sat;
```

```
% Elevasjonsvinkel til skjæringspunkt mellom kuleformet jord og flat jord
beta0 = acos(r/(r+h_sat)) + deg2rad(10)/2;  % Elevasjonsvinkel mot senter av hovedlobe
beta1 = acos(r/(r+h_sat));                  % Elevasjonsvinkel mot ende av hovedlobe

% Vinkel mellom satellitt og skjæringspunkt mellom kuleformet jord og flat
% jord, sett fra den kuleformede jordens senter.
gamma0 = beta0 + asin( (r+h_sat)/r*cos(beta0) ) - pi/2; % Vinkel mot senter av hovedlobe
gamma1 = beta1 + asin( (r+h_sat)/r*cos(beta1) ) - pi/2; % Vinkel mot ende av hovedlobe

% Elevasjonsvinkel til nytt jordplan
psi = 1/2*( gamma0 + gamma1 );

% Elevasjonsvinkel til senter av hovedlobe for flat jord
beta = beta0 - psi;

% Satellittenes høyde i det nye jordplanet
h_sat = r*sin(gamma0)*sin(beta)/cos(beta0);

% Vektor til senter av hovedlobe
% NB: Denne ikke er bestemt i x-retning, altså mot nord, ettersom dette er
% bevegelsesretningen til satellittene. x-koordinaten vil typisk være midt
% mellom satellittene. Den er derfor satt til 0, ettersom dette vil være
% typisk dersom satellittene står symmtrisk om origo, langs x-aksen.
p_xc_N = [ 0; h_sat/tan(beta); 0 ];

% Plasserer ut satellittene i på den flate jorden
p_B1_N = [  d/2; 0; -h_sat ];   % Posisjonsvektor [m] fra N til B1 sett fra N
p_B2_N = [ -d/2; 0; -h_sat ];   % Posisjonsvektor [m] fra N til B2 sett fra N

R_B1_N = eye(3);  % RKM fra B1 til N (vi roterer basisvektorene i N til å bli B1)
R_N_B1 = R_B1_N';
R_B2_N = eye(3);  % RKM fra B1 til N (vi roterer basisvektorene i N til å bli B1)
R_N_B2 = R_B2_N';

% Vektor fra satellitt til senter av hovedlobe
q_B1xc_B1 = R_N_B1*( p_xc_N - p_B1_N );
q_B2xc_B2 = R_N_B2*( p_xc_N - p_B2_N );

% Rotasjonsvinkler fra bodyramme til sensorramme på hver av satellittene
alpha1 = atan2( q_B1xc_B1(2), q_B1xc_B1(1) );
beta1  = atan2( q_B1xc_B1(3), norm( q_B1xc_B1(1:2) ) );

alpha2 = atan2( q_B2xc_B2(2), q_B2xc_B2(1) );
beta2  = atan2( q_B2xc_B2(3), norm( q_B2xc_B2(1:2) ) );

% RKM fra Sj til Bj (roterer basisvektorene i Bj til å bli Sj)
R_S1_B1 = Rz(alpha1)*Ry(pi/2-beta1);
R_S2_B2 = Rz(alpha2)*Ry(pi/2-beta2);

R_S1_N = R_B1_N*R_S1_B1;
R_S2_N = R_B2_N*R_S2_B2;

end
```

Funksjon – opplyst_omrade.m

```
function [ V1, V2, FV, L1, L2, X, Y, Z ] = ...
  opplyst_omrade( p_S1_N, p_S2_N, R_S1_N, R_S2_N, x_grid, y_grid )
% function [ V1, V2, FV, L1, L2, X, Y, Z ] = ...
%   opplyst_omrade( p_S1_N, p_S2_N, R_S1_N, R_S2_N, x_grid, y_grid )
%
% Finner felles hovedlobe til satellittene
%
% Input:
%  - p_S1_N: Posisjon satellitt 1
%  - p_S2_N: Posisjon satellitt 2
%  - R_S1_N: KTM fra S1 til N
%  - R_S2_N: KTM fra S2 til N
%  - x_grid
%  - y_grid
%
% Input:
%  - V1: Vinkel fra senter av hovedlobe til satellitt 1
%  - V2: Vinkel fra senter av hovedlobe til satellitt 2
%  - FV: Vinkel fra senter av felles hovedlobe
%  - L1: Avstand ut til gridpunkt for satellitt 1
```

```
%  - L2: Avstand ut til gridpunkt for satellitt 2
%  - X:  Meshgrid for x_grid
%  - Y:  Meshgrid for y_grid
%  - Z:  Meshgrid for 0
%

R_N_S1 = R_S1_N';
R_N_S2 = R_S2_N';

[ X, Y, Z ] = meshgrid( x_grid, y_grid, 0 );

V1 = zeros( size(X) );
V2 = zeros( size(X) );
FV = zeros( size(X) );
L1 = zeros( size(X) );
L2 = zeros( size(X) );

I = numel(X);
for i=1:I

    % Vektor mot mulig skipsposisjon
    p_x_N = [ X(i); Y(i); Z(i) ];

    % Propagasjonsvektor i N
    q_B1Ri_N = p_x_N - p_S1_N;
    q_B2Ri_N = p_x_N - p_S2_N;

    % Propagasjonsvektor i Sj
    q_B1Ri_S1 = R_N_S1*q_B1Ri_N;
    q_B2Ri_S2 = R_N_S2*q_B2Ri_N;

    % Vinkel inn på sensor
    V1(i) = abs( atan2( norm( q_B1Ri_S1(1:2) ), q_B1Ri_S1(3) ) );
    V2(i) = abs( atan2( norm( q_B2Ri_S2(1:2) ), q_B2Ri_S2(3) ) );

    if V1(i) >= V2(i)
        FV(i) = V1(i);
    else
        FV(i) = V2(i);
    end

    % Avstand ut til grid-punkt
    L1(i) = norm( q_B1Ri_S1 );
    L2(i) = norm( q_B2Ri_S2 );
end

end
```